AN ABSTRACT OF THE DISSERTATION OF

Kevin Green for the degree of Doctor of Philosophy in Robotics presented on August 31, 2022.

Title: Agile Bipedal Locomotion via Hierarchical Control by Incorporating Physical Principles, Learning, and Optimization

Abstract approved: ______________________________

Jonathan Hurst Ross L. Hatton

Robotic Bipedal locomotion holds the potential for efficient, robust traversal of difficult terrain. The difficulty lies in the dynamics of locomotion which complicate control and motion planning. Bipedal locomotion dynamics are dimensionally large problems, extremely nonlinear, and operate on the limits of actuator capabilities, which limit the performance of generic methods of control. This thesis presents an approach to the problem of agile legged locomotion founded on a first principles understanding of gait dynamics. This approach is built on the perspective that an understanding of locomotion is vital to the successful application of modern control and planning tools. We present 1) a ground-up analysis of legged locomotion as a dynamical phenomenon, 2) approaches that utilize dynamically meaningful reduced order models of locomotion, and 3) applications to the hardware robot Cassie via reinforcement learning.

Agile Bipedal Locomotion via Hierarchical Control by Incorporating Physical Principles, Learning, and Optimization

by

Kevin Green

A DISSERTATION

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Presented August 31, 2022
Commencement June 2023

Doctor of Philosophy dissertation of Kevin Green presented on August 31, 2022.

APPROVED:

Co-Major Professor, representing Robotics

Co-Major Professor, representing Robotics

Head of the School of Mechanical, Industrial and Manufacturing Engineering

Dean of the Graduate School

Kevin Green, Author

ACKNOWLEDGEMENTS

To my research mentors. Jonathan, thank you for teaching me to be an independent-thinking researcher and to keep the big picture in perspective. Ross, thank you for helping me build excitement, wonder and appreciation for the beauty of math and science. Alan, thank you for support and guidance as I ventured into a new field. To my mentors at the University of Michigan Medical School, Glenn Green, David Zopf and Kyle VanKoevering. Thank you for giving me the opportunity and the flexibility to contribute to extremely impactful projects and for having the patience to teach me the basics of biology and anatomy which I lacked. To C. David Remy whose passion for legged locomotion, mechanical engineering and scientific research inspired me to follow my path studying bipedal robots. I am eternally grateful that David responded to the email I sent him as a freshman, even though I had left the subject line blank.

To the students I have collaborated with, have mentored me and have taught me how to be a mentor, thank you. To Nils Smit-Anseeuw, Yevgeniy Yeslevskiy, and Wyatt Felt thank you for mentoring me throughout my undergraduate research career. The welcoming, supportive, thoughtful environment of the RAM-Lab had a huge impact on me. To Andy Abate, Taylor Apgar, and Patrick Clary for welcoming me to the Dynamic Robotics Lab and teaching me more than I could have imagined, faster than I could have ever imagined. To Helei Duan, Jeremy Dao, Mike Hector, and Brian Layng for working with me on research ideas that years earlier I thought were totally impossible. To Jonah Siekmann, John Warila, Yesh Godse, Ryan Batke, Fangzhou Yu, Andrew Sanders and Grace Stridick for working with me as I have learned to be a mentor. I hope I have been able to have a positive impact on your professional life and I thank you for trusting me with your time and effort.

To my family and friends for helping support me through the most challenging five years of my life. Through publishing rejections. Through technical failures. Through wildfires. Through political turmoil. Through a global pandemic.

Through the isolation of social distancing. Through the loss of my grandma and aunt. Thank you Anna Hua, Darren Cheng, Kari Green, Ted Xiao, and Melanie Green.

Thank you Mom and Dad, more than I can express.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

Page

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF FIGURES (Continued)

Figure Page

LIST OF TABLES

Chapter 1: Introduction

As robots find increased use to areas beyond the factory, they require increased mobility. Mobile robots are seeing application in a wide variety of applications such as autonomous aerial vehicles in surveying [9], autonomous underwater vehicles in ocean sampling [103], differential-drive wheeled vehicles in warehouses, and legged robots in construction site surveying [151]. These new applications require different physical capabilities. In particular legged robots hold promise for use in rough terrain and environments engineered to accommodate the human form factor. No robot is yet able to match or exceed the rough terrain traversal capabilities of animals or humans. To realize this potential legged robots must be reliable, robust, efficient, and agile.

This thesis presents a principled understanding of the fundamental principles of legged locomotion and a series of academic research efforts which examine specific methods of investigating these properties and applying them to bipedal robot control. The core principles are intended to be a useful guide to engineers and scientists new to the area of agile legged locomotion. While the technical implementation methods will evolve and iterate in the coming decades, I hope that these principles will be a more enduring understanding of the core phenomenon of legged locomotion which we hope to understand and implement.

1.1 A Brief History of Agile Legged Locomotion

While many walking machines were created before Raibert's, his hopping and running machines represent a leap forward in dynamism. In the early 1980s, Marc Raibert's Leg Lab at MIT built and demonstrated legged robots whose agility is impressive even today. These robots were able to hop, run, skip and trot through the use of pneumatic actuators and an external air source [124]. They even demonstrated front flips and backflips on 3D robots [115].

While pnuematically actuated legged robots have fallen out of vogue, the control principles Raibert and his lab pioneered are the foundation of much work even today [122]. The pneumatic hopping machines exploit the passive dynamics of an pnuematic piston to act as a nonlinear spring [121].

In addition to Raibert and contemporaries' work on hopping robots, passive dynamic walking robots emerged [105]. Tad McGeer built a demonstration of a completely passive walking machine which was able to sustain steady state locomotion down a slight slope [104]. This machine's gait was incredibly fluid and natural, even reminiscent of a human walking gait. Later, Andy Ruina's Biorobotics and Locomotion Lab at Cornell extended these passive dynamic walking principles to minimally actuated walking robots [33]. These robots generally use a small ankle push off to inject energy into the gait efficiently to make up for all the unavoidable losses in the mechanism and associated with foot touchdown. The most well known of these minimally actuated walking robotics is the Cornell Ranger robot, which is the most efficient walking robot to this day [15, 89, 34]. In a 2011 demonstration, this robot walked 40.5 miles unassisted on a single battery charge over the course of 30 hours and 50 minutes [134].

The spring-like passive dynamics of the Raibert hopping robots were extended to electric motor driven legged robots. Gill Pratt's continuation of the MIT leg lab showed electrically actuated walking robots such as Spring Flamingo, Spring Turkey and later M2 [119, 118, 73, 120, 131]. All of these robots used series elastic actuation, which is where the interface between the prime mover and the kinematic joint is intentionally made compliant. In Pratt's robots this is used for both impact tolerance and as a precise force transducer [131, 129, 130]. This design approach is in common usage today with robots such as ANYbotics' ANYmal [10] and Appronik's Draco [12, 5] which are built around modular series elastic actuators.

While compliance is extremely useful for accurate force control, it is also useful to directly enable the gait dynamics for a robot whose actuators cannot act transparently enough. Early hopping robots such as the bow legged hopper and the ARL monopod are good examples of using this type of passive dynamics in locomotion [24, 4]. Bipedal robots such as MABEL and ATRIAS use their passive springs to absorb impacts and create smooth force profiles which would not be possible with their relatively high inertia actuators [60, 77, 36, 128, 127].

An alternative approach to engineering passive compliance is to design a legged robot with low inertia actuation then implementing the leg impedance in software. One of the early proponents of this approach is Sangbae Kim from MIT with his Cheetah robots [138, 160, 18]. In addition to control bandwidth, he argues that the regenerative braking by backdrivable motors is vital to enabling efficient locomotion and real world usage [110]. The most extreme robot to demonstrate this design philosophy is the Minitaur robot developed at the University of Pennsylvania [90]. This robot's legs are four bar kite linkages where two links are directly driven by a brushless DC motor. This results in a robot with incredibly low inertia legs. In recent years this low gear ratio approach has become the defacto standard among a large set of commercially available quadrupedal robots.

The agility robotics robots Cassie and Digit sit between engineered passive compliance and low inertia actuation. Their legs have passive fiberglass springs that insulate the motors and their transmissions from foot impacts, but they also have low gear ratios for backdrivability and software compliance [11, 161]. This design approach has lead Cassie to demonstrate some of the most impressive performance by electrically actuated bipedal robots [132, 142].

1.2 Agile Legged Robot Control Methods

Bipedal locomotion requires feedback control of the unstable, underactuated, highly nonlinear dynamics. This control can range from quite simple heuristic, separated feedback controllers (Raibert control laws) pioneered in the MIT Leg Lab to extremely complex (provably stable hybrid zero dynamics). This section will summarize some of the most common methods of control. Many of the methods are intrinsically connected to motion and footstep methods but we will try to focus on the lowest level of whole robot level control.

1.2.1 Raibert Control Laws

The simplest control is perhaps that of the Raibert hoppers. The basic structure to Raibert control is to separate the problem of locomotion into three separate pieces with their own control laws. First, by increasing the pressure of the pnuematic leg cylinder during mid-stance the leg will push off harder, injecting energy into the system. Second, by changing the touchdown leg angle the robot can modulate its forward velocity. Finally, during stance the robot applies hip torque to stabilize the body pitch. These three control laws are essentially well tuned PID control laws which are only applied during specific parts of the gait cycle.

1.2.2 Virtual Model Control

An evolution of this control approach is virtual model control (VMC). This was first applied to spring turkey and later spring flamingo [73]. Virtual model control is all about applying virtual forces on the robot's body. It consists of two separate but important parts. First is how the desired body forces are chosen based on a series of virtual springs and dampers. If the robot is in single stance phase, virtual springs and dampers are applied to the height of the robot body and to the bodies rotation. This is known as the 'virtual granny walker' because it supports the robot's body but does not pull the robot forward. The forward position/velocity is not able to be controlled because of the underactuation intrinsic to single stance. However, if the robot is in double stance, it has an additional degree of control and is now able to control the forward velocity of the body. The forward force is calculated using a damper between the robot and a moving target, which is called 'the virtual dog track bunny.' The spring/damper analogy is useful to internalize what is happening but in practice these are simply PD controllers on the robot's body. This method of applying PD control to a robot's body is the standard practice in many bipedal control methods today [51, 11].

The second part of virtual model control is how the forces on the main body are created. These forces on the body have to be created using the stance leg(s) joint torques. This mapping of desired body forces to joint torques is done purely kinematically in this early work. This is sometimes called Jacobian transpose control because it is based around using the transpose of the Jacobian of the leg's kinematics to map body forces to joint torques. Jacobian transpose control is still used today, particularly in robots with very lightweight legs and heavy bodies [47, 73].

1.2.3 Whole Body Control

Many more recent control methods follow a similar structural approach to virtual model control but with increased fidelity. Replacing the virtual model 'granny walker' and 'bunny' are PD controllers tracking body motion from a motion planner. A common modification is to have the PD controllers act in acceleration space instead of force space. This generalized better as inertia changes with body configurations.

The other big improvement over virtual model control is modern whole body control. Many similar control methods go by different names such as operational space control, whole body control, or even simply inverse dynamics. These methods exploit the linearity of multibody dynamics. If we examine a generic form of the manipulator equation,

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = B\tau + J_c^T\lambda + J_f^T f \tag{1.1}$$

we can see the linearity in generalized coordinate acceleration ($\ddot{q}$), actuator effort (τ), kinematic constraint forces (λ) and ground reaction forces (f). In this expression M is the mass matrix, B is the velocity product forces, g contain the gravitational forces, and B maps the control effort into generalized coordinates. J_T is the Jacobian of the kinematic constraints and J_f is the Jacobian to the contact points which apply ground reaction forces. Further, if we are interested in controlling the acceleration of an arbitrary point ($\ddot{x}_T$) on the robot, that is linear in generalized accelerations as we can see in the expression

$$\ddot{x}_T = J_T \ddot{q} + \dot{J}_T \dot{q}. \tag{1.2}$$

This linearity allows us to solve for desired body accelerations by using standard linear algebra techniques while exactly accounting for the robot's internal dynamics [79]. Additionally, if we would like to impose constraints such as torque limits or friction cones we can form the acceleration matching task as an efficient quadratic program [11]. These whole body control methods are excellent in that they fully account for the instantaneous nonlinear dynamics and underactuation intrinsic to legged robots. However, their principle drawback comes from that instantaneous nature. They are not able to look ahead in the dynamics or through the gait cycles. It could be very useful to alter the reactions of the robot based on the point of the gait cycle. With whole body control any forethought must come from the motion planner.

1.2.4 Hybrid Zero Dynamics

A method that considers the whole of the gait when designing the feedback control is hybrid zero dynamics (HZD). The core principle of hybrid zero dynamics is about distilling down to the underactuated components of motion in such a way that these uncontrollable components are naturally stable. The process involves finding a nominal trajectory and set of holonomic constraints which can be enforced through the joint actuation. In a way, HZD is both a whole body controller and a reduced order model. The full order holonomic constraints are optimized in conjunction with the reduced, zero dynamics trajectory. A drawback is the limited nominal motions. To achieve variations on the gait (speed, step length, expressive features, ect.), a new trajectory must be optimized. Further, the space of gaits which have holonomic constraints which are provably stable is smaller than the space of gaits which are generally feasible. This method has shown applications on many bipedal platforms such as RABBIT [29], MABEL [146], ATRIAS [65] and DURUS [98].

1.2.5 Reinforcement Learning

While reinforcement learning has been used for legged robots for decades [150], it has recently dramatically increased in popularity and has demonstrated some extremely impressive, dynamic behaviors. These works include quadrupeds like Minitaur [149] and ANYmal [80], and bipeds like Cassie [161] and Digit [27]. Reinforcement learning for agile legged locomotion has mostly consisted of model-free, neural network based methods in simulation. In particular, many methods use a policy-gradient algorithm such as PPO, SAC, TRPO or DDPG [136, 62, 135, 97].

1.3 Thesis Format and Contribution

This thesis explores the principles of agile, efficient and robust bipedal locomotion and applies these principles to motion planning and reactive control. Beyond simply building more agile and robust controllers, we aim to find human understandable principles, strategies, and objectives.

Most of the chapters are adapted from published work which was produced as a collaboration between myself and other researchers. These chapters are marked with a title pages which includes the title, authors, publication venue and status if the work is an in review manuscript. Further, each of these works contains a paragraph describing each of the authors' contribution to the work and to the manuscript.

Part I contains a first principles description of agile, robust locomotion. This part articulates the perspective about the fundamental basis of legged locomotion and presents supporting evidence. It concludes with a more specific chapter examining how hierarchical control is an emergent feature of legged locomotion.

Part II looks at insights from, and planning methods that use, reduced order models of locomotion. Chapter 4 examines how to make aerial running robust to ground height disturbances. The novel input linking method allows us to optimize multiple trajectories for varying ground height that share actuator commands. This method showed that for a series elastic leg it is useful to extend the leg through the touchdown event and to swing the leg backwards at touchdown. This reinforces previous biomechanics and optimization studies which shows the utility of these strategies. Chapter 5 looked at a method for efficiently performing motion planning using an dynamic model which does not have a closed form solution. Many motion planning methods struggle with choosing between simple, mathematically efficient models and dynamically rich, expensive models. This work uses a data-driven approximation of the controlled first-return map which eases the computational burden at the expense of approximation error. However, there is nothing stopping the optimizer from extending the model well beyond the valid domain. For legged robots this corresponds to states and inputs which cause the robot to fall. The novel addition is to include a failure margin function which is an oriented distance function in state-action space. We show that this addition greatly increases the reliability of the planning method for an actuated spring loaded inverted pendulum model.

Part III looks at integrating knowledge of agile locomotion with the power of reinforcement learning to control real bipedal robots. Chapter 6 looks closely at how the distribution of environments which the robot sees in training can affect the resulting control policy. Specifically, we look at distributions of stairs compared to flat ground. The stairs policy is highly successful at climbing and descending flights of stairs in simulation and on hardware. This is *the first demonstration of a perception-free, unconstrained, bipedal robot climbing and descending stairs.* Most interesting is how we can examine the difference in the gait and control policy between the stair controller and the flat ground controller. Chapter 7 takes a step toward integrating reinforcement learned control policies into a control hierarchy. We aimed to train a control policy which takes commands from and is trained to imitate the motion of an optimized actuated spring loaded inverted pendulum model. This resulted in a controller capable of high quality walking and grounded running. Further, the controller showed signs of spring mass locomotion such as double humped ground reaction forces in walking and similar body oscillations. Chapter 8 looks to take this approach of using reduced order models to inform learned control and extend it to one-off maneuvers such as high speed turns. In particular we are interested in motions which rely on modulating angular momentum. To this end, we elevate from our inverted pendulum models to a single rigid body model with massless, ideal legs. These references are used to train learned controllers to perform high speed locomotion and high speed, abrupt turns.

As many chapters are adapted from self contained publications, they present their specific contributions in their body. The following are what I view as the most impactful contributions from the work contained in this thesis.

- A concise and approachable description of the first principles of agile locomotion.
- Specific evidence that swing leg extension and angular retraction produce a more robust response to ground variations.
- An approach to mitigate the risk of invalid solutions when planning with data-driven Poincarè models.
- The first demonstration of blind traversal of stairs by an unsupported bipedal robot.
- Evidence that learning locomotion on stair environments resulted in increased swing leg retraction compared to flat ground environments.
- The first demonstration of training a learned controller for a biped to emulate motions from a reduced-order model.
- The first example of learned, high-speed, agile turning maneuvers on an unsupported bipedal robot.
- A novel epilogue reward system for learning fixed duration maneuvers.

Part I

Fundamental Principles of Legged Locomotion

Chapter 2: Emergent Properties of Agile Legged Locomotion

In this chapter we will examine several important features of legged locomotion which make it unique from an engineering perspective. These features are built from the first principles of agile locomotion but we will go into more detail about how they relate to the current methods and approaches in use. This section should give unifying context for the following chapters' technical research. The primary intended audience is engineers and early career researchers in area of legged locomotion who would like to have a deeper understanding of the problems, difficulties and unique features of legged locomotion, independent from the engineering challenges and general challenges of robotics.

2.1 Gaits Can Be Described by Reduced-Dimensional Representations

Legged locomotion is, at its core, a high dimensional quasiperiodic behavior. Walking and running is a cyclical manipulation of body shape which produces a net displacement. To function in any useful context this locomotion cycle must be able to be adjusted from step to step. This could be to avoid obstacles, to speed up or slow down, or to adjust for variations in the terrain. The implicit question is how does this cycle change? I contend that while the cycle exists in a very high dimensional space (20-50 dimensions for legged robots and 100s of dimensions for biological systems) it is varied along a much smaller dimensional manifold. I ask forgiveness from the mathematicians and dynamicists for my casual usage of "manifold." The idea here is that the nominal gaits across different locomotion goals will smoothly vary and lay on a smaller dimensional surface. When disturbed (for example a push, a foot slip, or an unexpected ground height change), these orbits may deviate from the manifold but will very quickly be forced back onto the manifold. Over the next step or two the path on the manifold will converge back to the specific desired orbit. Also importantly, the description of a dynamical state projected onto the gait manifold should contain most of the information needed to adjust gait level control actions such as foot placement and energy injection.

Reduced-order models of locomotion are the most explicit embodiment of this dimensionality reduction in practice. For example, inverted pendulum models are a principled minimal description of a legged robot. They describe the center of mass dynamics of the robot and in particular the underactuation which results from a point foot contact. This is clearly insufficient to fully describe the dynamics of a legged robot, but it is an excellent starting point to understand the effect of foot placement and in some cases stance energy injection. Various models either prescribe the stance leg forces (inverted pendulum, linear inverted pendulum, spring loaded inverted pendulum) or can allow for some simplified descriptor of leg actuation (Raibert hoppers, actuated spring loaded inverted pendulum) [85, 95, 56, 123, 76]. More descriptive models such as centroidal momentum or single rigid body models extend this to a larger representation which allows them to perform more motions and encapsulate more of the robot's dynamics.

These models' usage is exactly in line with this smaller dimensional configuration space because they are used to understand the effect of control actions on the entirety of the gait cycle. These models are most powerful in control structures as the model in an online model predictive controller. They enable reasoning about the effect of varying footstep placement on the gait cycle several steps into the future. Importantly for their efficacy they reason about this effect within this reduced dimensional space. Then these planned motions are lifted into the full-order space of the robot and are stabilized with classical whole-body control techniques.

Learned control methods are a little more nebulous because it is extremely difficult to inspect the optimized controller, but I would not be surprised if many methods functionally converged to something resembling this reduced dimension understanding of gait cycles. We observe that bipedal learned walking and running controllers will generally converge to something that resembles a limit cycle in simulation regardless of disturbances or initialization. This limit cycle varies as we change dynamic parameters of the system and is different than the emergent cycle on hardware. This is true across speeds which may indicate there is a strong feedforward signal with a gentle feedback reaction. I believe this relates closely to the idea of existing on a smaller manifold of motions and could be a very useful feature of learned planners which encode the full observed robot state to a minimal form for planning.

2.2 Trajectories Are Emergent Behaviors from Multifaceted Gait Objectives

Motions are emergent results of the goal of locomotion, not the key interest. As engineers and scientists, we observe and judge robot gait quality by observing its trajectory. We see the poses, but more importantly we observe how they change over time to form an opinion about the motion. Humans are particularly good at observing, understanding, and even identifying specific individuals based on human gait [83, 147]. However, we are much worse at recognizing animal motions [32]. This raises an issue: if we use our aesthetic preferences to judge the quality of a robot's gait, we risk ensuring that our robots will move like humans. While this could be useful if the true goal is human perception, if we actually care about the robot's performance we need to understand the factors that created that human motion and emulate them but applied to the robot. Simply imitating "natural human gait" on a robot without actually getting the physics of locomotion correct misses the purpose. The gait must be driven by the integration of the physics of locomotion and gait objectives to see the benefits of natural, agile locomotion.

In the field of legged robotics we struggle with meaningful, objective comparisons of different control methods. While a cynical view is that this is used to cover up deficiencies of different approaches, a fairer view would be that there is fundamentally not a single measure of legged robot performance[1]. Instead, it is

[1]This is to say nothing of the difficulty in comparing different hardware platforms that may

complex, multifaceted, and situation dependent.

Injury avoidance is a vital cost function for locomotion. Risk of injury manifests in both falls and in damage caused by impacts during locomotion. Avoiding falls is an important interest which should be considered when designing hardware and choosing locomotion paths, motions and control strategies. Further, the risk of falling varies with the environment. A robot will have to consider footholds much more carefully when traversing difficult terrain such as a boulder field or stepping stones compared to walking across a uniform paved surface. In addition to the specific physical attributes of the environment, the uncertainty of the environment greatly impacts the risk of failure. If the robot's sensing is degraded because of lighting conditions or the environment itself has varying ground properties (e.g. a muddy trail) then the robot will need to be more conservative in its behavior to keep the risk of falling acceptably low.

Second is slightly less intuitive, the robot must avoid damage during normal locomotion. This can be thought of the difference between a human runner tripping, falling and tearing a ligament versus a runner developing shin splints over the course of a cross-country season. A robot's gait selection can affect the amount stress on different components which can cause premature failure.

One example of this second failure mode is shown in Fig. 2.1. This shows photos of the Cassie robot after a learned controller was run for a 5k race [41]. This controller emphasized heel touchdown with the hope of reducing impacts, energy loss and disruption to the gait. However, this caused the foot to slide as it was touching down and wore away the heel of the foot over the course of the 53-minute race. Further, the impacts were transmitted to a mechanical weak point in the motor crank which actuated the foot. This led to a failure where the crank cracked. The lesson here for gait design is that hardware design, control strategy, and gait design must be integrated. An example engineering tradeoff this prompts is when to modify the mechanical leg design to better accommodate the stresses, and when it is better to modify the gait to avoid long-term damage.

have particular challenges which suit different control approaches.

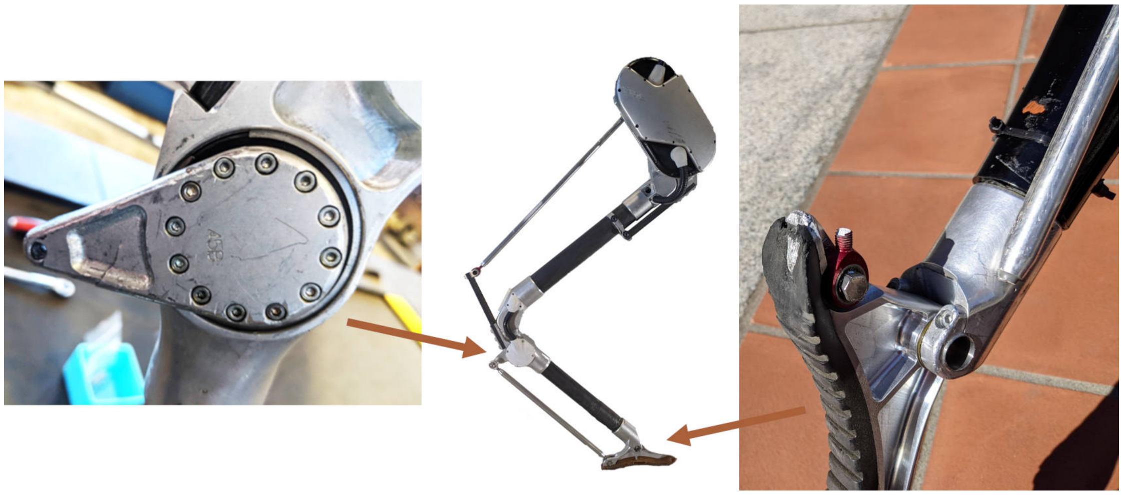


Figure 2.1: Two examples of damage caused by poorly designed gait motions. The gait in question was taught to heel strike which applied large impulsive loads to the foot actuation system, breaking the connecting rod and cracking the aluminum crank. It also caused the foot to slide at touchdown, wearing away the rubber of the foot.

A second class of objectives beyond self-preservation are the high-level mobility goals. At the most basic level there is the task goal, e.g. walk to the kitchen. This goal often is paired with a description of the urgency of motion. The urgency may be high, such as in a hot food delivery task, or it could be relatively slow, such as preemptive restocking. Another mobility goal could be the expressiveness of the gait [46]. For example, when delivering parcels in a residential setting the robot would want to be non-threatening, predictable and pedestrian. If the same robot was patrolling a secure site, their gait should be more authoritative and deliberate.

Finally, a legged robot must balance energy usage against all these other factors to maximize its battery life. Like in automotive design, energy efficiency is traded off against power, speed, capacity, and expressiveness. Much work has investigated optimizing robot gait for pure energy efficiency. This has produced some incredible smooth, efficient gaits but their robustness to unexpected disturbances is not considered at optimization time.

Humans and animals are very good, but not perfect at balancing these same objectives in a context dependent manner. As an example of risk adjustment, consider how a human will change their gait when walking through a room in the pitch dark compared to in the light. In the dark room a person will be more cautious, shortening footstep length, swinging legs more carefully and being closer to statically stable walking. However these balances are not perfect in humans, take for example competitive running. Runners would like to minimize the risk of injury and to that end require coaching from experts to refine their technique.

2.3 Touchdown Is the Most Challenging Instant of the Gait and Greatly Influences Design and Control Choices

When walking or running, one of the largest sources of disturbances and uncertainty is the ground itself. In general, touchdown is a collision event which results in extremely abrupt changes in system velocity. These sharp changes in velocity can make feedback control near touchdown difficult and often requires either loose feedback gains or very careful, contextual gain design [165]. While it is possible to sense or estimate features of the ground surface, this is difficult to do so reliably and precisely due to conditions changing from foothold to foothold. This includes properties such as ground height, ground compliance, friction properties and ground normal. This disturbance works differently than the traditional disturbance sources often considered in control theory. Touchdown disturbance appears suddenly and substantially at the instant of touchdown, even changing the moment that touchdown occurs.

Compounding on the manifestation of disturbances is the fact that touchdown delineates a loss of control authority. As previously discussed, dynamic legged locomotion is underactuated, meaning that there are components of the robot's state which cannot be influenced directly at an instant in time. As an example, one of these components is the angular momentum around the contact point during stance. To have a realistic, useful robot we need to control the long-term behavior of these state components. The way many of these vital components are controlled is through foot placement. If the robot has too much forward momentum, the correct reaction is to place the foot further forward. The instant after a foot is placed on the ground, there is a long length of time before the robot can place the next foot to influence the future motion of the robot.

To achieve robust locomotion, gait strategies must be selected with consideration to how they interact with touchdown disturbances. The motion and control strategies should seek to minimize the long-term disruption on the gait from these disturbances. The nominal motion at touchdown can be designed in such a way that it reduces the sensitivity of the gait to variations. A great example of this phenomenon is swing leg retraction in drop steps. When running humans and birds swing their legs, they extend them further forward than their ideal leg touchdown angle, then swing them backwards to the ideal touchdown angle [20, 157]. If the ground is higher than expected they will touchdown with a more angled leg, and if the ground is lower, they will touchdown with a more vertical leg. This has several effects on the gait which are desirable. First, it modulates the maximum leg force during stance so that the leg is not overloaded and risk damage. Second, it automatically regulates the body height of the next step by adding or removing forward momentum. On step ups, the more forward foot placement lifts the body height while removing forward momentum. On step downs, the more retracted foot placement lowers the body and accelerates it forward. This strategy is useful because it is much more important to immediately regulate body height to avoid falling or kinematic problems in leg swing, then to adjust forward speed in subsequent steps.

2.4 Actuator Dynamics and Limitations Dictate the Features of Effective Gait

From an engineering perspective, muscles are a very complex actuator. Skeletal muscle exhibits length dependent force exertion, velocity dependent force and even different types of skeletal muscle contractions [44, 126]. These muscle properties are extremely important to understand human and animal gait. Studies which attempt to model human gait or to optimize prosthesis assistance often need to use extremely high dimensional models with individual muscle models [140, 66]. As we look toward legged robots, it makes intuitive sense that our engineered actuators' dynamics and limitations are very different but are just as important to the robot's gait.

The actuator dynamics associated with a robot are vitally important to enable effective gait. When I describe actuator dynamics, I am referring to the physical properties and the actuation limitations of a primal mover of a legged robot. By far the most used actuation method in dynamic legged robots currently is brushless DC (BLDC) motors with transparent speed reductions. Transparency is an extremely important attribute of an actuator in the context of locomotion. Dynamic transparency is an actuator's ability to admit, sense and respond to external forces and impacts. BLDC motors are particularly well suited to legged robots because of their high power to weight ratio, high efficiency, and high torque density [91]. Even with the high torque density, robot legs often require more torque than is possible with direct-drive systems without the motor size becoming unwieldly or overdriving the windings to the point of unreliability. Further, the legs do not require the top speeds possible with a direct-drive brushless motor. This leads to the intuitive engineering solution of a mechanical speed reduction: a geartrain. An important question is once you are adding a transmission, how much do you reduce the speed of the motor? One approach to choosing a transmission ratio is to do it based on the speed and torque required for a target walking gait. This is the classic mechanical engineering method, one that is used when selecting a drive motor for a lathe or an industrial fluid pump. How fast should it go and how much torque will it need to provide at different speeds? This is an excellent method for tasks which require continual, steady state effort like machine tools or industrial equipment. However, the requirements for walking and running gaits are much more than a speed and a force. They require dynamic accelerations under different load conditions, continual feedback for balance and adjustment and acceptance of impacts and the variations of impacts. Looking holistically at the requirements for gait, it makes sense that when selecting the transmission ratio that the dynamic transparency is an important factor. Transparency is reduced when the transmission ratio is increased because the reflected inertia of the rotor and the reflected damping increase with the square of the ratio.

When considering the transparency of the actuation, the balance of factors is pushed more towards low gear ratios and more backdrivable transmissions. This can be seen in many agile and dynamic legged robots which have entered the market. The MIT mini cheetah quadruped is extremely agile, able to run at high speeds and perform dynamic maneuvers like backflips [1]. This robot uses BLDC motors with a low, six to one ratio planetary reduction [87]. The Agility Robotics' bipedal robot Cassie uses BLDC motors with a 16 to one reduction on the sagittal plane hip and knee joints [133]. This robot has performed some of the most impressive traversal feats in the real world, ranging from blind stair traversal to setting the bipedal robot 5k record to setting the bipedal robot 100 meter dash record [41].

2.5 Physical Requirements for Leg Design Are Unique and Demanding

The features and requirements for an agile robot leg are very different from robot arms or other mechatronic systems. The core requirements are high force capacity, high unloaded acceleration, accurate position control and the ability to stabilize the body in stance. The leg will need to be able to apply a significant fraction of the total robot's body weight, be that over half for a biped or over a quarter for a quadruped. This requires the leg able to support and accelerate its own weight as well as that of the body which includes batteries, compute hardware, perception and any functional loads (manipulators, advanced sensors, ect.). This last requirement is less specific because there is a very interesting continuum of ways this can be satisfied ranging from extremely high bandwidth force control to carefully engineered passive dynamics with low bandwidth control. Using purely active control, the leg's actuators must have high bandwidth because the foot may only be on the ground for several hundred milliseconds at a time. In this time the foot much establish ground contact, sense the disturbances that may have been introduced then apply the corrective forces to stabilize the robot. An excellent example of a robot that use this approach is the MIT mini cheetah [160]. Alternatively, this can be accomplished by having a leg with very well designed passive dynamical properties that result in a much smaller required actuation bandwidth. Examples of robots which use this second approach are SALTO, ATRIAS, and Raibert's original hopping robots [64, 74, 122].

Agile, high-speed locomotion requires quick leg repositioning. The limiting factor in this is often not the maximum speed of the leg but the acceleration capability of the leg. It must lift off the ground after being swept backwards in stance, reverse direction, swing beyond the desired touchdown angle, then retract toward the planned touchdown location. One of the most intuitive things that limits legged runners (human, animal and robot) is how quickly they can get their legs repositioned. When a runner is moving too quickly and cannot get its leg far enough forward in in time, it will tumble forward and fall to the ground. Beyond being able to physically reposition the leg in time, the robot must be able to precisely position the foot at touchdown to stablize and control the overall forward and lateral momentum.

Further the leg must be able to accommodate impacts without damage, gait disruption, or excessive energy loss. The reasoning behind this necessity was discussed in Section 2.3.

Together these requirements make for a strong leg with high force density, high dynamic transparency and low overall weight. One of the most important features is to reduce the task space inertia as much as possible. The task space inertia is the measure of the apparent inertia at the foot of the robot. It includes the mechanism of the leg, the physical inertia of the leg components and the actuator reflected inertia, as they translate to the foot. It is direction dependent, describing how much force is required to accelerate the foot in a particular direction. A much more detailed discussion of the measure, how to calculate it, and the implications for leg design can be found in [2]. The principle ways that task space inertia can be reduced are through the transmission ratios, the mass distribution and the leg kinematics. Reducing the transmission ratios while maintaining torque requirements would require physically larger motors, but this will result in a more transparent system. The exact balance is a trade off as the larger motors increase total mass and will generally increase the resistive losses in the motor for a given torque output. The task space inertia can also be decreased by moving mass to be as proximal as possible on the leg mechanism. Finally, the leg kinematics are very important to the foot's apparent inertia. The more the joints are able to spread out the velocity required post impulse the lower the inertia will be. This means that extended legs are very bad for inertia, which makes sense intuitively to humans. When falling a distance, people must avoid having their knees locked because that dramatically increases their foots task space inertia. Their knees and hips are unable to reduce the impulse by contracting so the impulse needed to slow their body is transferred up the leg, likely causing damage to bones.

Chapter 3: Hierarchical Control Is Not a Forced-Compromise but Instead Is an Implied Structure

Hierarchical control is a property of many of the most popular control approaches for legged robots. The hierarchy of control methods serve to separate components that require different execution speed, different model fidelity and different amounts of world information. An interesting question is if this is merely a necessary compromise or if it is emergent from the structure and principles of legged locomotion. It could be argued that this is a simplification because of computational limitations. If in the future we have access to orders of magnitude more computation power onboard a robot, it might become possible to plan a precise, optimal full over trajectory that is verifiably robust to disturbances at thousands of Hertz. I argue, that if that was an option it would fundamentally not be necessary and may not ever produce better performance than a well built hierarchical control structure which can actually be implemented. This is because legged locomotion actually has layered, distinct levels of action. These range from very fast stabilizing feedback, to more abstract foot placements, to long-term body motions. A diagram of the difference levels of the hierarchy are shown in Fig. 3.1.

At the lowest level and fastest rates we have passive dynamics of the robot. All active control, even if it is running at an extremely high rate is limited by the physical bandwidth of the actuators. In most legged robots (other than piezo-electric and direct drive robot) this is a significant physical delay to react to disturbances. This means that the physical properties of the robot define the highest rate reaction. By engineering passive compliance into a robot leg, the high rate reactions can be prescribed. Moving up a level, low-level control like whole-body control can run at very high rates, with very accurate models of the robot's dynamics and kinematics. They are able to do this because they have very short or non-existent forward horizons. In other words, they don't think about the future, they

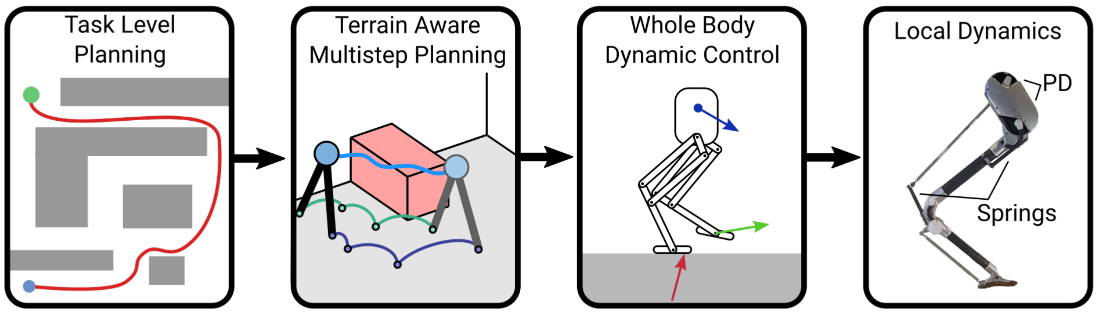


Figure 3.1: Hierarchy of control for legged locomotion. The levels of the hierarchy are separated by their execution rate, by the fidelity of their internal model and by the horizon for which they look ahead.

are purely reactive. At a middle level there is a short horizon, terrain aware step planner. This is necessary for traversal around obstacles in the environment and maintain dynamic balance. This requires a forward horizon over which to consider the effect of actions. However, this level of planning does not require a sophisticated, full-order model or nearly as fast of an execution rate. Instead much simpler models such as SLIP, IP, LIP or even purely kinematic models can be sufficient for basic locomotion. Finally, the highest level considered here is the large scale path planning which can operate slowly on very simple models. These models can be as simple as a differential drive $SE(2)$ model in a 2D world. The world model can be relatively coarse but could need be quite large depending on the task.

From a biomechanics perspective we know that humans and animals use something akin to this hierarchical control. At the lowest level, legs have engineered passive dynamics through the skeletal system and muscle-tendon units. These passive dynamics allow the leg's muscle groups to efficiently produce the forces necessary in stance while accommodating disturbances without destabilizing the walker. Slightly higher, there are instinctive reactions which operate with relatively low latency from the brain stem. Above that humans will plan out a series of footsteps and a rough sense of the body movements. We specifically know that people only need need two step lengths of visual information to plan footholds

[101]. Further we know that people limit their step planning to between 1.5 and 2 seconds into the future [102]. It is interesting that a different study showed that the amount of time people spend looking at the footholds increases as the terrain becomes rougher [111]. This may imply that for selecting footholds, that even if you have the capability to sense and plan footsteps further in the future it would not be helpful. Higher than this humans plan their gross motions through space, not considering the specific gait physics on how they will get there.

Part II

Multistep Planning through Dynamically Meaningful Reduced Order Models

Chapter 4: Direct Optimization of Open-loop Disturbance Rejection

This chapter contains a study which investigates optimization of robust gait plans across varying environments. It contains a novel approach to input linking the different scenarios together. The resulting motions independently arrive at strategies that are observed in animal locomotion.

Contributions

Kevin Green wrote the manuscript, devised, implemented and performed the numerical experiments. Jonathan Hurst and Ross L. Hatton supervised the investigation and co-wrote the manuscript.

Planning for the Unexpected: Explicitly Optimizing Motions for Ground Uncertainty in Running

Kevin Green, Ross L. Hatton, and Jonathan Hurst

IEEE International Conference on Robotics and Automation
Paris, France (Remote)
May 2020

4.1 Abstract

We propose a method to generate actuation plans for a reduced order, dynamic model of bipedal running. This method explicitly enforces robustness to ground uncertainty. The plan generated is not a fixed body trajectory that is aggressively stabilized: instead, the plan interacts with the passive dynamics of the reduced order model to create emergent robustness. The goal is to create plans for legged robots that will be robust to imperfect perception of the environment, and to work with dynamics that are too complex to optimize in real-time. Working within this dynamic model of legged locomotion, we optimize a set of disturbance cases together with the nominal case, all with linked inputs. The input linking is non-trivial due to the hybrid dynamics of the running model but our solution is effective and has analytical gradients. The optimization procedure proposed is significantly slower than a standard trajectory optimization, but results in robust gaits that reject disturbances extremely effectively without any replanning required.

4.2 Introduction

Dynamic locomotion such as running and walking has many dimensions beyond position trajectories, which are merely one symptom of the resulting behavior. As such, new approaches are needed to incorporate powerful existing motion planning and control methods with the dynamic behaviors of legged locomotion. Complicating factors include underactuation, nonlinear hybrid dynamics, large system dimensionality and significant uncertainties in ground properties. However, legged locomotion is not so complex as it first appears, because most behaviors can be described by relatively simple reduced-order models, showing some promise for planning within this dynamic space. Many reduced-order models consist of a point mass body and a massless leg that can apply forces from a contact point toward the point mass, where body motion is only influenced by gravity and the forces applied by the leg. Examples of this type of model include the inverted pendulum (IP) model, the linear inverted pendulum (LIP) model, the spring loaded inverted pen-

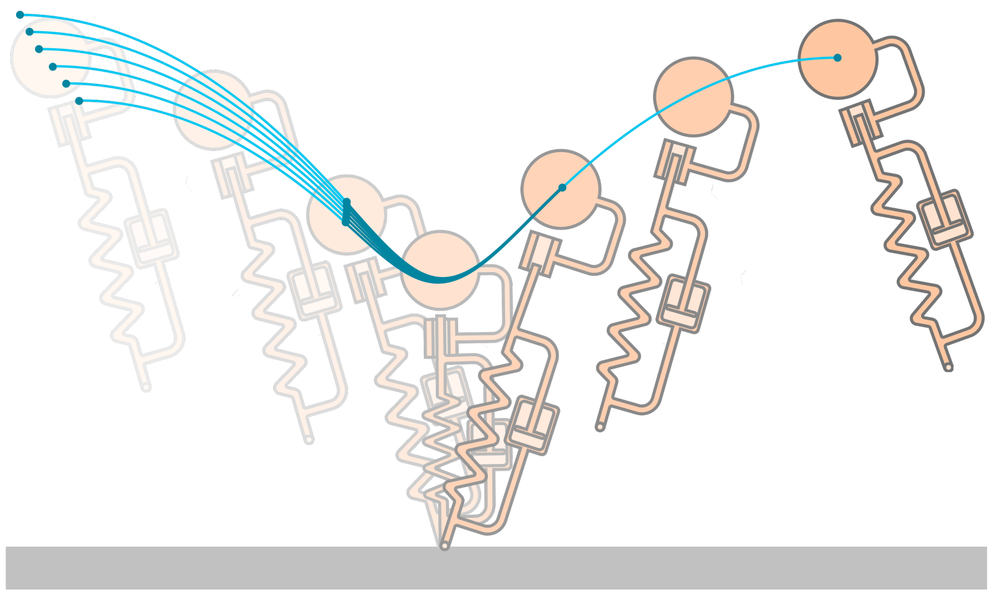


Figure 4.1: A robust motion plan for the actuated SLIP model. All trajectories use the same control inputs, yet they all converge from different heights to the same final apex state.

dulum (SLIP) model, and the actuated spring loaded inverted pendulum (ASLIP) model. The differentiating factor between these models is the calculation of the applied leg force.

When walking and running, the ground height for the next step cannot be measured perfectly so intrinsic robustness to errors in ground height is extremely desirable. Ground sensing is a difficult problem because of the complex dynamics of legged locomotion[25]. Feet can slip which complicates proprioceptive estimation [19]. Cameras, LIDAR and other sensors experience difficult-to-predict motion throughout the gait cycle. Due to the sharp cost of failures (falls and subsequent damage) it is desirable to be robust to the uncertainties in ground height. This is the perspective that motivates much of the work on generating robust blind locomotion [125].

This work describes a method to search for open loop actuation plans for the ASLIP model that exactly reject a range of disturbances in the ground height

(Fig. 4.1). There are multiple reasonable ways to define rejection of a ground height disturbance; here we define rejection as returning to the desired apex state relative to the disturbed ground height. If a different desired behavior can be expressed in terms of the final states in the disturbance cases (e.g. return to the desired apex state relative to the previous state), then the method presented here enables investigation of that behavior. The resulting open loop plans for this model consist of swing leg motion and a trajectory for the extension of leg actuator. Leg swing motion controls the foot's touchdown angle and the leg actuator controls the set point of the damped spring in the leg. Our approach is to optimize the expected motion for a set of disturbance cases in one large problem. This is not trivial because it requires linking the inputs between the different cases including accounting for the timing of the hybrid transitions. The solution we propose for input linking is effective and includes analytical gradients throughout.

We provide an overview of relevant existing work in Section 4.3. The ASLIP model and its hybrid dynamics are defined in Section 4.4. In Section 4.5 we describe the two trajectory optimization techniques we are comparing and the simulation we use to test performance. The results of both optimizations and the simulation testing are reported in Section 4.6. Closing remarks are in Section 4.7.

4.3 Background

Studying reduced order models has provided insight into the phenomenon of legged locomotion and how to create dynamic walking and running in robots. The passive SLIP model explains most of the effects observed in human ground reaction forces during running and walking [56]. If this model is extended with swing leg dynamics, it can generate all common bipedal gaits [55]. Further, the SLIP model can be used to generate foot placement policies that regulate forward speed [49]. In the actuated and damped ASLIP model, an open loop cyclic reference trajectory was shown to reject both ground height and ground impedance variations when hopping vertically [78]. Preflex (pre-reflex) behaviors on an extension of the ASLIP model can aid in mitigating sensing delays when the system encounters disturbances [75].

These discoveries inform us on how to create and stabilize legged locomotion.

The insights from reduced order models have been successfully leveraged in generating control methods for legged robots. The ATRIAS robot successfully demonstrated robust blind walking by leveraging knowledge from reduced order models about foot placement, energy injection and clock based open-loop feed-forward signals [74]. The RHex hexapod robot was stabilized with a feedback policy from a clock-driven SLIP [8].

Other control approaches directly manipulate reduced order models to plan motions online in a model predictive control approach. A common approach for bipedal locomotion is it use the linear inverted pendulum model due to its linear dynamics [50, 45, 11]. This allows the robot to quickly reason about where to place its feet given the estimated state and the goals from a high level planner or an operator.

4.4 Dynamic Model

The ASLIP model consist of a point mass body with mass m and no rotational inertia as well as a massless leg. This leg has a linear extension actuator that is assumed to be a rigid position input. The output of this actuator is connected to the massless point foot through a damped linear spring with stiffness k and viscous damping b. This system has states corresponding to the body position x, y and velocity $\dot{x}$, $\dot{y}$, the leg actuator set point position r_0 and velocity $\dot{r}_0$, and the passive spring deformation r_p. The leg actuation extension is limited to be between l_0 and $l_0/2$ where l_0 is used as a descriptive length of the leg. Gravitational acceleration is g. These coordinates and parameters are labeled on the system diagram in Fig. 4.2. We include the actuator velocity as a state because we use the acceleration of the set point $\ddot{r}_0$ as the control input. The commanded acceleration is limited in magnitude as a proxy for absolute torque limits. Here $5g$ is used as the maximum acceleration The passive spring deflection must be a part of the state because during flight phase the spring and damper create first order dynamics. This model has both a discrete control action in the foot placement as well as a continuous

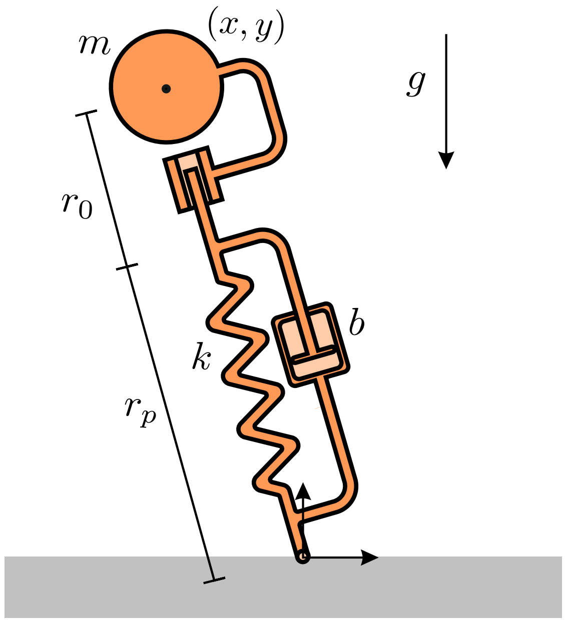


Figure 4.2: The ASLIP model in stance with labeled parameters and state variables. The origin for the body position (x, y) is the contact point.

control action in the leg extension actuator which results in a much richer action space than the passive SLIP model.

4.4.1 Equations of Motion

This system has two distinct dynamic modes, flight and stance. In flight phase the dynamics of the body and the leg are fully decoupled. The body exhibits ballistic motion with the equations of motion

$$\ddot{x} = 0 \tag{4.1}$$

$$\ddot{y} = -g. \tag{4.2}$$

The leg set point motion is only influenced by the commanded acceleration ($\ddot{r}_{cmd}$) because it is assumed to be rigidly actuated,

$$\ddot{r}_0 = \ddot{r}_{cmd}. \tag{4.3}$$

The passive deflection of the spring has a first order response during flight due to the lack of foot mass,

$$\dot{r}_p = -\frac{k}{b} r_p. \tag{4.4}$$

In stance phase the toe is constrained to stay in contact with the ground and leg is able to apply forces on the main body, but only in the leg length direction. In the application of this constraint, the spring deflection variable (r_p) is made dependent on the body and set point position. To simplify the description of the dynamics, the origin of the body's position coordinate system is set to be the foot contact point. This makes the spring deflection and velocity

$$r_p = r - r_0 \tag{4.5}$$

$$\dot{r}_p = \dot{r} - \dot{r}_0, \tag{4.6}$$

where r is the total leg length and $\dot{r}$ is the total leg velocity. The stance center of mass dynamics are

$$\ddot{x} = \frac{xF}{mr} \tag{4.7}$$

$$\ddot{y} = \frac{yF}{mr} - g \tag{4.8}$$

where F is the leg force on the body, defined as

$$F = k(r_0 - r) + b(\dot{r}_0 - \dot{r}). \tag{4.9}$$

We assumed that the set point is a rigid position source so it is influenced only by the $\ddot{r}_{cmd}$ control signal despite the external loads it is supporting in stance.

4.4.2 Hybrid Transition Model

We are using this model to represent the sagittal plane dynamics of bipedal aerial running. This means that the robot will cycle through the phases of flight, left leg stance, flight and right leg stance before repeating. Given symmetries in the sagittal plane we can analyze only half of this cycle. The start and end points of this half cycle are somewhat arbitrary, but a common choice is to use the apex condition in flight as the start and end point [56]. With this start and end point, the phases of our half cycle are descending flight, stance, ascending flight.

Touchdown is when the model transitions from flight to stance which occurs when the foot reaches the ground. This is more precisely described as the states at the moment of touchdown intersect the guard surface

$$\sqrt{x^2 + y^2} = r_0 + r_p. \tag{4.10}$$

Liftoff is where the model transitions from stance phase to flight phase when the ground reaction force goes to zero. This is not the same as when the spring has zero deflection because of the damping in the spring. The liftoff transition guard is described as having zero force in the spring

$$k(r_0 - r) + b(\dot{r}_0 - \dot{r}) = 0. \tag{4.11}$$

This model does not include the case where the force vector exits the friction cone and causes the foot to slip.

4.4.3 Nondimensionalization

If this model was being used to generate motions for a specific robot, one would use meaningful physical parameters from that robot for the reduced order model. Here we are interested in results that are generally applicable so we nondimensionalize our model. The nondimensionalization of our ASLIP model is based on previous work nondimensionalizing SLIP models and variations on SLIP models [71, 31].

	Symbol	Value	Description
Base Units	m	1 $[m]$	Mass
	l_0	1 $[l_0]$	Max set point Length
	g	1 $[g]$	Gravitational acceleration
Parameters	k	20 $[mg/l_0]$	Leg spring stiffness
	b	0.89 $[m\sqrt{g/l_0}]$	Leg spring damping
States	x	- $[l_0]$	Horizontal position
	y	- $[l_0]$	Vertical position
	$\dot{x}$	- $[\sqrt{gl_0}]$	Horizontal velocity
	$\dot{y}$	- $[\sqrt{gl_0}]$	Vertical velocity
	r_0	- $[l_0]$	Leg set point length
	$\dot{r}_0$	- $[\sqrt{gl_0}]$	Leg set point velocity
	r_p	- $[l_0]$	Leg spring deflection
Inputs	$\ddot{r}_{cmd}$	- $[g]$	Leg set point acceleration

Table 4.1: Nondimensional system parameters, states and inputs for the actuated SLIP model.

The characteristic units are the maximum leg set point length, mass of body and gravitational acceleration. All parameters and states are represented relative to these quantities and are summarized in Table 4.1. Two parameters must be chosen, the leg stiffness and the leg damping. The numbers we selected are similar to previous SLIP modeling papers [31] and are based on observations of human biomechanics [7]. The leg stiffness we used is 20 $[mg/l_0]$ and the leg damping is 0.89 $[m\sqrt{g/l_0}]$. The damping value is such that the body and leg system has a damping ratio of 0.1 in stance.

4.5 Methods

As a standard of comparison we first present a conventional minimal effort trajectory optimization. Then we describe the explicitly robust trajectory optimization method we propose. Finally, we describe a separate testing simulation to analyze the disturbance rejection capabilities of the open loop motion plans produced by these two optimization methods.

4.5.1 Minimum Effort Optimization

The minimum effort optimization seeks to find a single state trajectory and input signal that will result in moving from an initial apex state to the following apex state while minimizing a measure of actuator effort. This is formulated as a direct collocation problem. In this technique, the optimizer has access to both the discretized states and control inputs as decision variables as well as the time between the discretizations. It is conventional to have the states evenly spaced in time with a single duration decision variable. The dynamics are imposed as constraints between subsequent states and their inputs based on numeric integration techniques. In this work we use trapezoidal integration. Much more detail on this approach can be found in [88]. Direct collocation optimizations such as those implemented here contain hundreds or thousands of decision variables. This means that it is not feasible to report every individual constraint. Instead we describe the general form of constraints which are applied across the time discretized states.

Our optimization is complicated by the three separate dynamic phases. Each phase of the dynamics is implemented with its own set of discretized state and input variables and phase duration variable. The final state of one phases is constrained to match the first state of the next phases. Additionally the final states of the first flight phase and the stance phase must be at the hybrid transition guards described above in Section 4.4.2.

The initial height and forward velocity as well as the final height and forward velocity are constrained to match user-specified values. To ensure the initial and final states are apex states, the vertical velocities are constrained to be zero.

The objective (J) we use is the integral of the set point acceleration squared,

$$J = \int_0^\tau \ddot{r}_0^2 \, dt \tag{4.12}$$

where τ is the total duration of the motion. This is a useful objective both theoretically and practically. If this was a real system where the set point actuator is a geared electric motor, this objective is proportional to the thermal energetic losses in the motor due to accelerating the actuator inertia [166]. Practically this smooths the acceleration commands which increases the accuracy of the trapezoidal integration scheme [88].

The constraints and objective functions are generated using a modification of COALESCE, a MATLAB based optimization problem generation library [84]. We generated the constraints, constraint Jacobian, objective and objective gradients analytically while preserving their sparsity. This produces a nonlinear programming (NLP) problem that can be solved to local optimality using an off the shelf nonlinear solver. We used IPOPT (an interior point technique package) to solve the NLP, but other implementations or optimization techniques could be used [158].

4.5.2 Disturbance Aware Trajectory Optimization

The method we use to optimize for variations in ground height is an extension of the minimum effort technique presented above. We not only create all the state variables and inputs for the expected motion, but also for some number of disturbance cases with different initial conditions. All of the disturbance cases are thought of as different versions of what may happen during the planned step; this means that they must all share the same set point motion. This is ensured through an input linking opteration described in the following section. Each disturbance case still has the same final state constraints, forcing the optimizer to try to funnel each of the disturbance cases to the single final state.

The minimum effort objective was removed for this trajectory optimization because it was found to prevent the convergence of the optimization. This makes this optimization problem more accurately a constraint satisfaction problem. The limits on the maximum acceleration of the set point ensure that even without an explicit objective that the motion of the set point is relatively smooth.

One subtle aspect of this problem is that the disturbance cases are able to each select a different leg touchdown angle. This may appear to conflict with the assumption that the model cannot know which disturbance case is occurring, but the optimizer is implicitly selecting a leg swing trajectory. Each disturbance case contacts the ground at a different time at a different leg angle. This set of leg angles over time exactly constitutes a leg swing retraction policy.

The difficult aspect of working with all the disturbance cases together is that their control inputs must be linked together. The system will not know which of the disturbance cases it is in so the inputs as a function of total time must match. Each disturbance case has flight and stance phases that take different lengths of time, so we cannot rely on the indexing of the collocation nodes to link instances. This problem exists when trying to link disturbance cases in any system that goes through hybrid transitions or has a variable duration.

The solution is to use an additional set of decision variables evenly spaced through time that are not linked to any specific dynamic phase or collocation node. Each of the collocation node inputs are constrained to be equal to the linear interpolated value from these control points.

To describe this constraint and its gradient, we first define a generic linear interpolation function and a zero order hold function. The linear interpolation function (with extrapolation) as it is conventionally understood is

$$\mathrm{LI}(x, v, x_q) = \begin{cases} \frac{x_2 - x_q}{x_2 - x_1} v_1 + \frac{x_q - x_1}{x_2 - x_1} v_2 & x_q \leq x_2 \\ \frac{x_{i+1} - x_q}{x_{i+1} - x_i} v_i + \frac{x_q - x_i}{x_i - x_i} v_{i+1} & x_i < x_q \leq x_{i+1}, \\ & 2 < i < N-2 \\ \frac{x_N - x_q}{x_N - x_{N-1}} v_{N-1} + \frac{x_q - x_{N-1}}{x_N - x_{N-1}} v_N & x_{N-1} < x_q \end{cases} \tag{4.13}$$

where $x \in \mathbb{R}^N$ is a strictly increasing vector of the sample points, $v \in \mathbb{R}^N$ is the values of those sample points and $x_q \in \mathbb{R}$ is the query point. For use in the gradient expression we need the zero order hold function (with extrapolation),

$$\mathrm{ZOH}(x, v, x_q) = \begin{cases} v_1 & x_q \leq x_2 \\ v_i & x_i < x_q \leq x_{i+1}, \\ & 2 < i < N-1 \\ v_N & x_N < x_q \end{cases} \tag{4.14}$$

were x, v, and x_q are the same as in the linear interpolation function definition.

Consider the kth collocation node with input u_k at time t_k, and control points described by time $T \in \mathbb{R}^m$ and value $U \in \mathbb{R}^m$. Our constraint (g) takes the form

$$g(u_k, t_k, U, T) = u_k - LI(T, U, t_k) = 0. \tag{4.15}$$

This constraint means that the the actual input u_k must be equal to the interpolated input from the control points at the current time t_k.

The optimization method we use benefits from having analytical gradients of all constraints, which we can describe for this function. The gradient of the constraint in 4.15 can be found using basic calculus. With respect to some decision variable (y) the gradient[1] of this constraint is

$$\begin{aligned} \nabla_y g(u_k, t_k, U, T) &= \nabla_y u_k && (4.16)\\ &+ \mathrm{ZOH}(T_1^{m-1}, \frac{U_2^m - U_1^{m-1}}{T_2^m - T_1^{m-1}} \nabla_y t_k, t_k) && (4.17)\\ &+ \mathrm{ZOH}(T_1^{m-1}, \frac{U_2^m - U_1^{m-1}}{T_2^m - T_1^{m-1}}, t_k) \mathrm{LI}(T, \nabla_y T, t_k) && (4.18)\\ &+ \mathrm{LI}(T, \nabla_y U, t_k) && (4.19) \end{aligned}$$

This constraint allows us to link all of the set point acceleration profiles together in a differentiable way. Unfortunately the gradient is undefined at the node points themselves and is frequently discontinuous. This discontinuous gradient can slow the optimization procedure but the constraints converge well to the desired tolerance in our application. A better option could be to use a piecewise cubic interpolation method to ensure the gradients are well formed, but it would decrease the sparsity of the constraints with respect to the decision variables.

[1] In this expression the symbol U_a^b is the vector of components described by $[U_a, U_{a+1}, ..., U_{b-1}, U_b]$.

4.5.3 Testing Simulation

To objectively test the disturbance rejection of the motions produced by the two trajectory optimization methods we implement a hybrid simulation of the ASLIP model. The simulation uses the same model dynamics and hybrid transitions except that we treat the set point trajectory (position $r_0(t)$ and velocity $\dot{r}_0(t)$) as the inputs. The system is forward integrated using MATLAB's variable step size ODE solver ODE45 with event sensing for the hybrid transitions.

Additionally, the leg touchdown angle must be defined for each of the motions. The robust optimization finds an explicit time varying leg touchdown angle. The minimum effort optimization only selects a single leg angle. One option would be to just use that single leg angle. A better option is to use the heuristic that the leg touchdown angle tracks a fixed horizontal touchdown location on the ground. This policy is similar to what guinea fowl do when they encounter an unexpected step up or down [21], and should ensure that the minimum effort optimization is not unfairly hindered with an unreasonable leg angle policy.

We run this forward simulation for a set of initial condition disturbances until it reaches the next apex state. If the disturbance is poorly handled it is possible that the body does not ever reach the next apex state. This generally is because the model falls into the ground before lifting off or reaches liftoff with a negative vertical velocity.

4.6 Results

To evaluate the optimization methods we generate motion plans for 625 sets of initial states and final goal states. The experiments were run single threaded on a standard desktop computer with an Intel Core i7-7700k and 24 GB of RAM. These correspond to every combination of five initial heights, initial horizontal velocities, final heights, and final horizontal velocities. This results in creating motion plans for steady state gaits, changes in speed, planned step ups and planned step downs. Observations on the optimization process and the resulting motions are presented for both the minimum effort and the explicitly robust optimizations. Finally, the performance of the plans from both methods is tested using a separate simulation for significantly more disturbance heights than were explicitly optimized.

4.6.1 Optimization Results

The minimum effort optimizations converged to optimality relatively quickly and reliably. The mean solutions time was 0.90 seconds and 95% solved in under 3.1 seconds. An example solution is shown in Fig. 4.3. The system lands with the leg set point slightly retracted and stationary. Then throughout stance the leg extends and reaches the maximum extension just before liftoff. As the leg extends it does positive work against the spring, replacing the energy lost in the damper during stance. During the flight phases the leg is smoothly retracted back to prepare for the next touchdown event. The ground reaction forces appear very similar to those seen in human running trials as well as in passive SLIP models.

In the explicitly robust optimization we used $+0.10$, $+0.05$, -0.05 and -0.10 $[l_0]$ as the errors in ground height for the disturbance cases which means this problem has five times the number of variables and constraints of the minimum effort problem. The mean solutions time was 4.3 seconds and 95% of successful solutions solved in under 8.9 seconds. This is notably slower than the minimum effort optimization, particularly when you consider the average time per iteration which was 0.01 seconds for the minimum effort and 0.2 seconds for the robust optimization. This is to be expected because of the over five times difference in number of decision variables and constraints between the two problems.

An example solution to the robust optimization is shown in Fig. 4.4. We can see in the top plot that all five initial heights converge back to the nearly same final height and forward velocity. Each trajectory has a different touchdown angle, becoming steeper for later touchdowns. The second plot shows that as the different disturbance cases touchdown, the leg is already extending in length. All the trajectories lift off at different points in time as the leg reaches its peak extension which is well short of the maximum extension of 1 $[l_0]$. In the ground

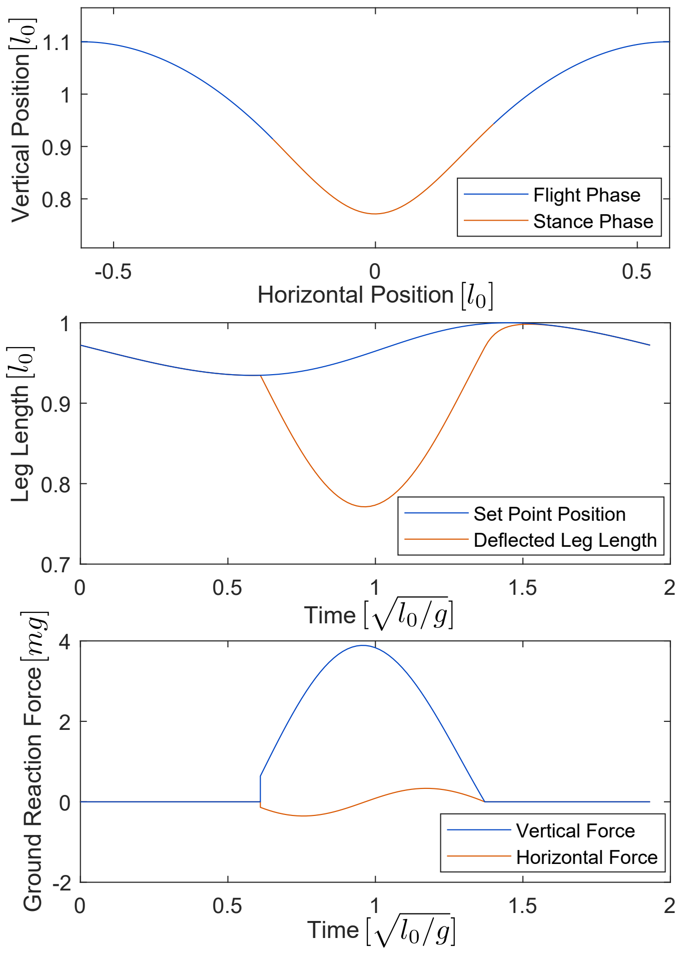


Figure 4.3: An example solution from the minimum effort optimization. We see that the model lands with the leg retracted, then smoothly extends to the maximum length at lift off to replace the energy lost in the leg spring damping.

reaction force plot at the bottom, we can see that the later the touchdown, the greater the peak force vertical force. This intuitively makes sense because it should require a larger vertical impulse to reverse the vertical velocity of the body and return it to the final desired height.

4.6.2 Simulation Testing Results

The performance of the two different trajectory generators was tested using the simulation described in section 4.5.3. Each trajectory was tested using eleven different vertical disturbances representing different step ups and step downs, between $+0.1$ $[l_0]$ and -0.1 $[l_0]$. An example result is shown in Fig. 4.5 for a steady state gait. Looking at the minimum acceleration results in blue, we can see that touchdown points are at similar heights because the leg set point has almost zero velocity at this point in the cycle. As these motions exit stance, they have drastically different forward velocity but a smaller range of final heights compared to the initial disturbances. The robust trajectories in orange show a very different reaction. The touchdown states are in a much tighter grouping due to the leg extension and the precise leg placement policy generated by the optimization. The states converge through stance and ascent until they reach the apex state. All of the robust apex states in this figure have less than 0.001 $[l_0]$ error in final height and 0.001 $[\sqrt{gl_0}]$ error in final forward velocity.

Looking at the results of all of the simulations we observe that 14% of the disturbances caused the minimum acceleration policy to fail to reach a valid subsequent apex state because the body contacted the ground before it reached an apex state. None of the tested disturbances caused the robust policy to fail. When comparing the final state error of the conditions where the minimum acceleration policy did not fail, the robust policy had on average 43 times less height error and 81 times less velocity error.

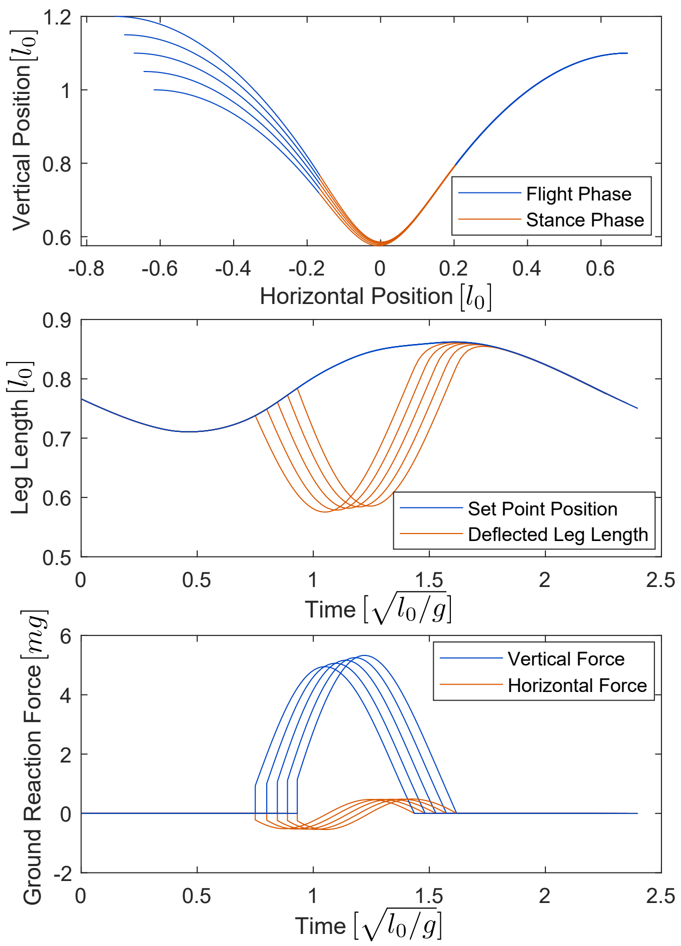


Figure 4.4: A robust motion plan for the actuated SLIP model. All trajectories use the same control inputs, yet they all converge to the same final apex state.

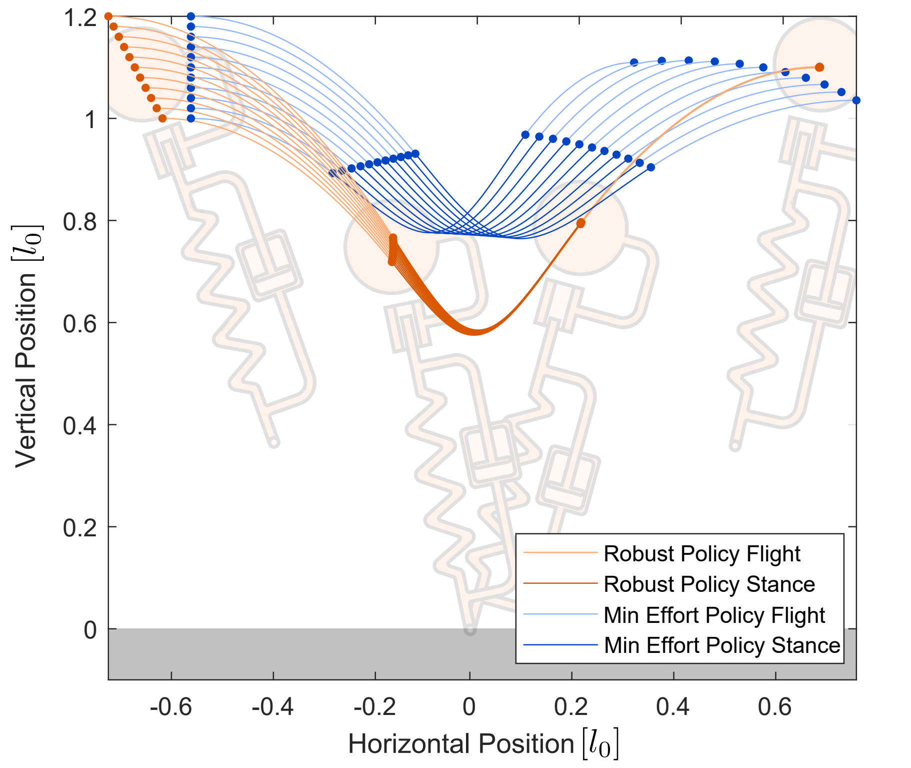


Figure 4.5: Simulation results of the open loop plans for the robust policy and the minimum effort policy. The robust policy (orange) has an apex height error of less than 0.001 $[l_0]$ and an apex velocity error of less than 0.001 $[\sqrt{gl_0}]$ for all disturbance cases. The swing leg of the robust policy is a product of the optimization while the minimum effort policy uses ground speed matching.

4.7 Conclusions

We presented a new method to create open loop plans for an ASLIP model that are extremely robust to ground height uncertainty. The presented approach consists of optimizing many different trajectories for different disturbance cases while linking the inputs together. The input linking used here is effective and efficient due to its analytical gradients and can be applied to other hybrid or variable duration trajectory optimization problems across a set of disturbances. The results show that the robust open loop motions plans produced have an order of magnitude less final state error compared to the minimum effort plans.

Chapter 5: Real-time Motion Planning for Systems Without Closed Form Solutions

This chapter contains a study which investigates the performance of a motion planning method for systems with periodic motions and dynamics which have no closed form solution. In particular, it looks into how to handle the reality of catastrophic failure of these models. It proposes a novel failure margin function to help the optimizer avoid extrapolating the learned dynamic model outside of the viable space.

Contributions

Kevin Green wrote the manuscript, devised, implemented and performed the numerical experiments. John Warila contributed to the implementation and analysis of the numerical experiments and co-wrote the manuscript. Jonathan Hurst and Ross L. Hatton supervised the investigation and co-wrote the manuscript.

Footstep Planning for Agile Models of Locomotion using Failure Margin Constraints

Kevin Green, John Warila, Ross L. Hatton, and Jonathan Hurst

(Accepted)
IEEE International Conference on Robotics and Automation
Philadelphia, Pennsylvania, USA
May 2022

5.1 Abstract

When legged robots perform dynamic locomotion, their underactuated dynamics require motion planning to intelligently adjust footstep locations. Often bipedal footstep and motion planning uses mathematically simple models, such as the linear inverted pendulum, instead of dynamically-rich models that do not have closed-form solutions. We propose a real-time optimization method to plan for dynamical models that do not have closed form solutions and experience irrecoverable failure. Our method uses a data-driven approximation of the step-to-step dynamics and of a failure margin function. This failure margin function is an oriented distance function in state-action space where it describes the signed distance to success or failure. The motion planning problem is formed as a nonlinear program with constraints that enforce the approximated forward dynamics and the validity of state-action pairs. For illustration, this method is applied to create a planner for an actuated spring-loaded inverted pendulum model. In an ablation study, the failure margin constraints increased the amount of valid solutions by between 11% and 51% for different objectives and lengths of horizon. While we demonstrate our method on a canonical model of locomotion, we also discuss how this can be scaled to data driven models and full order robot models.

5.2 Introduction

The full-order dynamics of legged robots are computational infeasible for real time motion planning so motion planners generally use simplified, reduced-order models of locomotion. Two of the most commonly used models are the linear inverted pendulum (LIP) model [52] and the centroidal dynamics model [37]. These models balance the benefit lower dimensionality while still capturing the core underactuated dynamics of locomotion. Some of these models also have computationally attractive dynamics, particularly the closed-form dynamics of the LIP model.

Unfortunately, there is a trade-off to using some of these highly computationally efficient reduced order models. Many of these models fail to exhibit features of locomotion which are known to relate to robust, efficient locomotion and are observed in human and animal locomotion. Energetically optimal gaits for bipedal robots can include the center of mass oscillations vertically and step frequency that varies with speed [145]. Neither of these features appear in conventional linear inverted pendulum footstep planning. Further the spring loaded inverted pendulum (SLIP) model, not the LIP model replicates the center of mass dynamics and ground reaction forces of human walking and running [56], as well as the disturbance response of drop steps in guineafowl [21].

Considerable interest recently has focused on using these dynamically rich models for bipedal locomotion planning. Unfortunately, many more complex models have the problem that they are nonintegrable and analytical approximations are often unwieldy [72, 167]. An extreme approximation of the actuated spring mass model was shown to be extremely robust for low speed bipedal locomotion [11]. Recent work planned footsteps with the simple LIP model then mapped those plans to a more dynamic and complex actuated SLIP model [164]. A different approach is to approximate the step-to-step dynamics directly instead of approximating the continuous dynamics. This has been shown to be useful in a one-step optimal controller [16] and in a multistep model predictive controller [168]. One area that has not been as well investigated is handling failure conditions of these models. Models of legged locomotion have failure modes where they never return to an apex condition, primarily falling into the ground or from the foot slipping. These failure modes result in complex, non-convex failure boundaries in state-action space [69]. Naively bounding the state and action ranges as to not include any failures severely limits the dynamism of potential motions.

In this work, we propose an optimization planning method for cyclic locomotion of analytically intractable models that include complex failure conditions. Our method is based on use of a data-driven, differentiable approximation of the controlled (Poincarè) first return map and of a state-action failure margin function, shown in Fig. 5.1. Our failure margin function represents the signed distance to the failure boundary. This allows us to add constraints on the motion planning

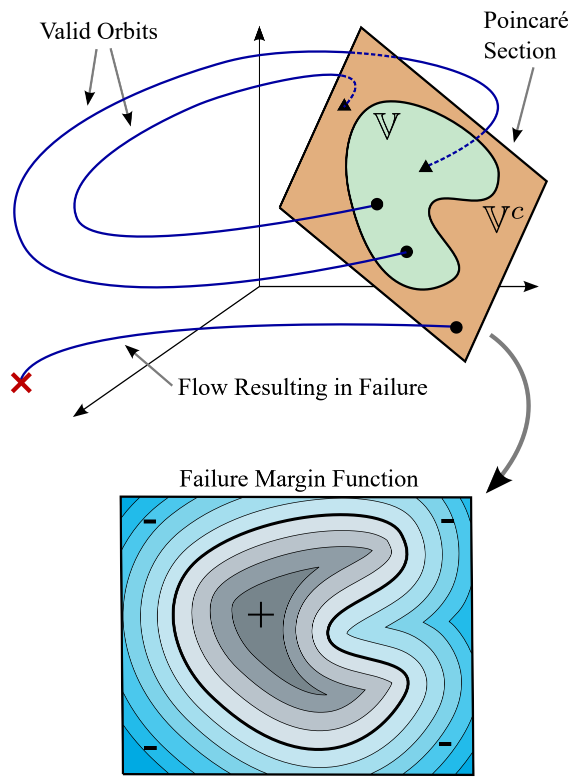


Figure 5.1: The relationship between the Poincarè section of a dynamical system and its failure margin function. Dynamical systems, such as legged locomotion, exhibit orbits which are effectively characterized with a Poincarè section. Legged locomotion is plagued with failure conditions (such as falling over) so we classify points on the Poincarè section into a set ($\mathbb{V}$) if they produce valid orbits. This set is then used to generate an oriented distance function that we call the failure margin function.

problem that ensures the each step in the state-action sequence is dynamically viable. Without this margin function we show that the optimization often chooses solutions that attempt to extrapolate into regions where failures occur. Section 5.3 defines the controlled first return map and failure margin function for a generic dynamical system. Section 5.4 describes how we generate approximations of the return map and failure margin function. A sample motion planning optimization problem is described in Section 5.5. We demonstrate these methods on an actuated SLIP model in Section 5.6. Finally, we provide areas for future extensions and improvements in Section 5.7.

5.3 Modeling of Periodic Gaits

To plan multiple steps ahead for a legged robot we need a dynamic model of locomotion. This model can be a simplified model, a full-order simulation of a robot, or even data from the real world robot. Full order simulations and real-world robots present additional challenges, because of their large state dimensions as discussed in Section 5.7.

Our dynamic model can be described by its configuration and velocity $\{\mathbf{q}, \dot{\mathbf{q}}\} \in TQ$, and has forward dynamics

$$\ddot{\mathbf{q}} = f(\mathbf{q}, \dot{\mathbf{q}}, \mathbf{u}), \tag{5.1}$$

where $\mathbf{u}$ is the control action. These dynamics are likely hybrid, but for notational clarity we will assume they are continuous.

5.3.1 Poincarè Map Discrete Dynamics

Legged locomotion is at its core a cyclic motion of the joints that results in net displacement. A powerful perspective to analyze cyclic motions is the Poincarè section and map. We define a surface of section Γ which is transverse to the flow of our state and which intersects with all gait trajectories of interest,

$$\Gamma \subset TQ. \tag{5.2}$$

This section is one dimension smaller than the full state and can be parameterized by a new, reduced state coordinate $s \in \Gamma$.

We seek to define a function, $\Phi(\cdot)$, which represents the controlled first return map. Our system is an actively controlled system, so instead of a traditional Poincarè Map, ours is augmented by the control actions. This map will allow us to predict the future state of the system by adjusting control actions (e.g. foot placement). The dynamical system may have continuously varying actions throughout its orbit, so to rein in the infinite dimensional function space we define a set of basis functions for a tractable parameterization. We define the space of all parameterized actions as $\mathbb{A}$. However, we must consider the fact that some state-action pairs will fail. For example, the ground reaction forces could violate the friction cone, cause the foot to slip and the robot to fall. This will result in a system trajectory that never returns to the Poincarè section. We can define the set of valid state-action pairs that will successfully return to the section as

$$\mathbb{V} \subset \Gamma \times \mathbb{A}. \tag{5.3}$$

Thus, our controlled first return map is

$$\Phi : \mathbb{V} \longrightarrow \Gamma. \tag{5.4}$$

which maps valid state-action pairs to the next state on the surface of the section.

However, the definition of this function is unhelpful to planning motions. Calculation of this function requires numeric integration due to the nonlinearity of the dynamics. Further the numeric integration gives us no method of finding the Jacobian of the transition function ($\nabla_{s,a}\Phi(s, a)$) reliably without slow, computationally expensive numeric differentiation.

5.3.2 Failure Margin Function

We would like to be able to easily identify if a candidate state and action will fail, and if it does, determine what direction we should move it to be closer to being a success. To achieve both of these, we construct a failure margin function which is an oriented distance function [43] (also called signed distance function), often used in computer graphics [99, 14]. To define this function, we employ distance function from a point (x) to a non-empty set ($A \subset \mathbb{R}^N$) with an associated norm ($\|\cdot\|$),

$$d_A(x) = inf\{\|x - y\| : y \in A\}. \tag{5.5}$$

we're going to pull information from our problem to populate we can define our failure margin function as an oriented distance function in terms of the valid set and the complement of the valid set,

$$b_{\mathbb{V}^c}(s, a) = d_{\mathbb{V}^c}(s, a) - d_{\mathbb{V}}(s, a). \tag{5.6}$$

Here the compliment of $\mathbb{V}$ represents the space of invalid state-action pairs, defined as $\mathbb{V}^c = (\Gamma \times \mathbb{A}) \setminus \mathbb{V}$. This function will be positive if the state-action pair is valid and negative if the pair is invalid. Its magnitude represents the minimum distance to the boundary of success and failure.

5.4 Approximation of the Step-to-Step System

To facilitate real-time, optimization based planning methods we develop a fast to evaluate, differentiable approximation of the controlled first return map and failure margin function. The failure margin function allows the optimization to constrain the state, action pairs to be in the valid set, $\mathbb{V}$. Efficient differentiability of our approximation is extremely important because we would like to be able to use gradient-based optimization algorithms such as sequential quadratic programming or interior-point methods.

5.4.1 Controlled First Return Map Approximation

We use a standard supervised learning method to fit our controlled first return map approximator to a training data set. Many structures of approximator could be used, such as polynomials and Gaussian processes but we use feed-forward neural networks. The approximation of the true first return map, $\Phi(s, a)$, is referred to as $P(s, a)$. The training data set is generated by sampling uniformly state-action pairs and performing the numeric integration. This will either fail, in which case we discard the sample, or it succeeds, in which case we add the initial state, action and final state to the data set.

5.4.2 Failure Margin Function Approximation

The failure margin function is more difficult to create an approximation of because it cannot be directly sampled. To generate data for our failure margin function we need a relatively accurate method of calculating the oriented distance function from (5.6).

We decided to sample the distance function using a direct application of our definition of the margin function from (5.6). We uniformly sample the state-action space, classify each point as either valid or invalid and build two sets of points. We then construct two k-d trees, one for valid points and one for invalid points. If we have a sample point, these trees allow us to efficiently find the closest valid or invalid point. Now we can sample our oriented distance function. We select a random point in state-action space, sample if it is a valid or invalid point by forward simulating. Now we look to calculate our best approximation of (5.6). If our point is a valid state-action pair then $d_{\mathbb{V}}(s, a) = 0$ and if it is invalid then $d_{\mathbb{V}^C}(s, a) = 0$. Now we only need to calculate the distance to the remaining set which is the canonical use-case for our kd-tree. This allows us to build a training set which we can use to fit the approximation (M) to the sampled oriented distance function such that, $M(s, a) \approx b_{\mathbb{V}}(s, a)$.

5.5 Footstep Planning Optimization Problem

There are many ways to formulate a footstep planning problem depending on the required behavior. Here we use a basic formulation which tasks the system with reaching a commanded state N steps in the future. The problem is provided with s_0, the predicted next apex state, and s_{goal} the commanded goal state. This is an effective formulation for traversal of nominally flat, obstacle free environments with a higher level planner or human commanding target states.

The footstep planning problem is formed as a nonlinear programming problem of the following form.

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & h_n(x) = 0, \qquad n \in [0, N-1] \\ & g_n(x) < 0, \qquad n \in [0, N-1] \\ & h_{\text{goal}}(x) = 0. \end{aligned} \tag{5.7}$$

This optimization problem finds a sequence of states and actions of length N that are dynamically consistent and reach the final goal state while minimizing the objective $f(x)$. The decision variable x represents the next N states and actions.

$$x = [s_1, s_2, ..., s_N, a_0, a_1, ..., a_{N-1}] \tag{5.8}$$

The objective function,

$$f(x) = \sum_{i=0}^{i=N-1} (s_i - s_{i+1})^T H (s_i - s_{i+1}), \tag{5.9}$$

minimizes the squared, weighted distance between sequential apex states which incentivizes gradual acceleration or deceleration. Inspired by excellent results of a similar planner on Cassie [11], we also will test the performance of this motion planning problem without an objective, i.e. $f(x) = 0$. Removing the objective resulted in significantly faster and more reliable convergence to acceptably smooth motion plans.

The approximated forward dynamics are enforced through an equality constraint,

$$h_n(x) = P(s_n, a_n) - s_{n+1}. \tag{5.10}$$

The viability of state-action pairs is enforced through an inequality constraint,

$$g_n(x) = M(s_n, a_n) + \epsilon, \tag{5.11}$$

which ensures the learned failure margin function is greater that a given threshold value, ϵ. Finally, the goal constraint enforces that the final state of the optimization matched the commanded goal state (s_{goal}),

$$h_{\text{goal}}(x) = s_N - s_{\text{goal}}. \tag{5.12}$$

We can implement the Jacobian of the constraints analytically through backpropagation of the approximations, $P(\cdot)$, $M(\cdot)$.

5.6 Illustrative Application

To test the utility and feasibility of this type of planner we choose to test it on one of the simplest systems that demonstrates the problematic features inherit to legged locomotion. These difficult features include nonlinear dynamics with no closed form solution, hybrid transitions, failures states and parameterized, low-level control policies. The simple, illustrative model we use is the nondimensionalized, actuated spring-loaded inverted pendulum (aSLIP) model. This model consists of a point mass body and a massless leg which can apply forces through ground contact. The leg consists of a damped spring in series with an extension actuator which is controlled throughout stance phase and can be instantly repositioned during flight phase. Detailed dynamics of this model and its hybrid transitions are presented in [59].

The configuration of the model is described the the horizontal and vertical

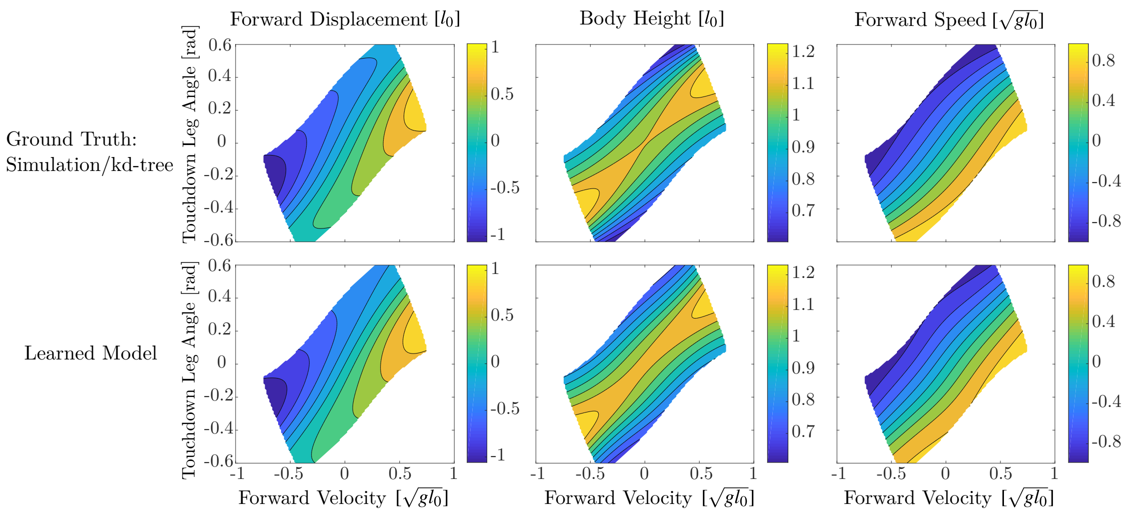


Figure 5.2: Comparison between ground truth and the approximation of the controlled first return map. For this figure we fixed the initial body height to $1.05[l_0]$ and the midstance leg extension to $0.05[l_0]$. We see that our approximator is highly accurate for most of the space, with the largest error near the limits of the input space.

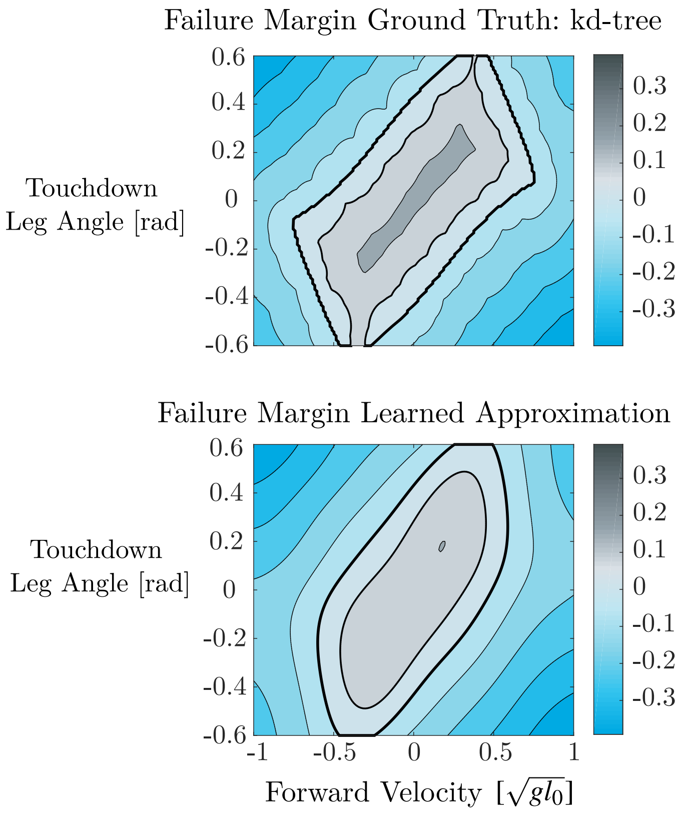


Figure 5.3: Comparison between ground truth and the learned approximation of the failure margin function. For this figure we fixed the initial body height to $1.05[l_0]$ and the midstance leg extension to $0.05[l_0]$. The learned approximation captures the general shape of the failure margin function but does not capture all the details of the zero contour (bolded contour line).

position of the body, $q = [x, y]$, while the velocity is the horizontal and vertical velocity of the body $\dot{q} = [\dot{x}, \dot{y}]$. This model has failure modes where it falls over or when the foot slips from a friction cone violation. The inputs are the leg angle at touchdown (α) and the motion pattern of the leg extension actuator which is in series with the damped leg spring. We select a simple actuator pattern inspired by the pneumatic Raibert hopping robots [122]. The leg actuation is parameterized by a single value (ΔL) that represents how far the leg actuator will extend at the maximum compression of the leg during stance. One could use a smooth extension profile, however for simplicity we chose to instantaneously change the actuators position (and thus the resting length of the leg spring) at maximum compression. This highlights a powerful feature of this method, one could design virtually any actuation profile or parameterization that they prefer. The action could be the true amount of energy to inject or remove, it could be the commanded amount of total energy in the system post extension, or it could be carefully designed to achieve increased robustness such as was shown in [59, 156].

5.6.1 Step-to-Step Approximation of the aSLIP Model

For the aSLIP model we selected the apex of flight phase as our surface of section. This is where the vertical velocity is zero during flight phase,

$$\Gamma = \{[x, y, \dot{x}, \dot{y}] \text{ s.t. } \dot{y} = 0 \wedge \text{Flight Phase}\}. \tag{5.13}$$

This allows us to represent the apex state by the reduced coordinates $s = [x, y, \dot{x}]$. Further we can exploit translational invariance to allow us to eliminate the horizontal displacement (x) from our apex state for the input. We need to know how far each step will translate the robot forward, but the starting location does not change anything about the step or its dynamics. This means the final state-action space will be

$$[y, \dot{x}, \alpha, \Delta L] \in \Gamma \times \mathbb{A}. \tag{5.14}$$

We define absolute limits on the state-action space based on reasonable gaits. These limits are

$$\begin{array}{rlcccl} 0.8 & [l_0] & \leq & y & \leq & 1.2 \quad [l_0] \\ -1.0 & [\sqrt{gl_0}] & \leq & \dot{x} & \leq & 1.0 \quad [\sqrt{gl_0}] \\ -0.6 & [\text{rad}] & \leq & \alpha & \leq & 0.6 \quad [\text{rad}] \\ -0.05 & [l_0] & \leq & \Delta L & \leq & 0.15 \quad [l_0]. \end{array} \tag{5.15}$$

The first return map is

$$[\Delta x_{i+1}, y_{i+1}, \dot{x}_{i+1}] = \Phi(y_i, \dot{x}_i, \alpha_i, \Delta L_i), \tag{5.16}$$

where Δx_{i+1} is the change in horizontal position from apex i to apex $i+1$. This is the first return map that we sample uniformly to create the training data set, form the kd-tree and sample the margin function. The kd-tree and margin function were created using a weighed 2-norm as their distance function with weighting matrix of: $\text{diag}(0.063, 0.250, 0.309, 2.50)$. This weighting was chosen based on the maximum range of each coordinate in the state-action space.

We tested several different neural network architectures, varying the width of the hidden layers and the choice of activation functions. The neural networks are optimized with the goal of minimizing the weighted 2-norm of the prediction error. The weighting matrix was also chosen to normalize the ranges of the different variables, $\text{diag}(0.250, 6.25, 0.25)$. The neural networks are implemented using PyTorch and optimized using the ADAM (adaptive momentum estimation) method with the default learning rate of 0.001 [92]. They were optimized until stationary which took approximately 100,000 iterations which corresponded to 5 to 10 minutes using a computer with an NVIDIA GTX 1080 and an Intel i7-7700k. We found that there was little difference in the performance between tanh and ReLU activation functions. Performance improved as the network size increased up to layer widths of 64 neurons. This led to the choice to use networks with 2 hidden layers of 64 neurons each with ReLU activation functions.

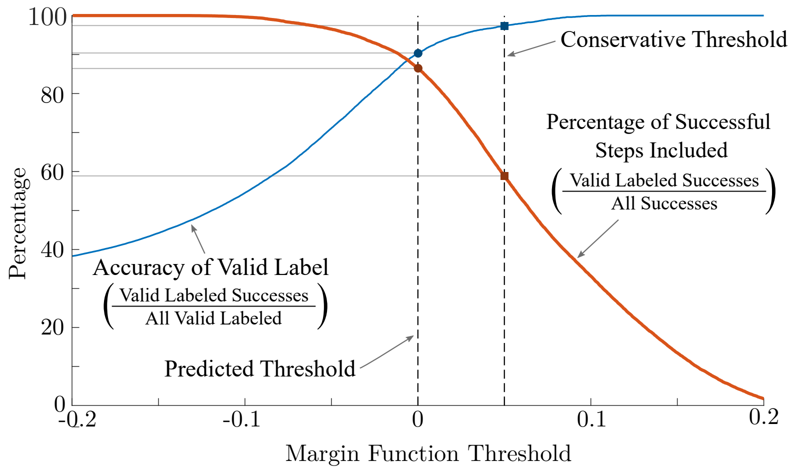


Figure 5.4: Performance of the failure margin function as we vary the valid point threshold value. We can see that as the threshold increases that the accuracy of the valid label increases. However, increasing the threshold also decreases the percentage of all successful steps that are included. Ideally, we want both of these to be as high as possible. We can adjust the threshold to balance the restriction of possible motions with confidence in accuracy of labeled points.

We can examine our approximation's performance compared to ground truth with the contour plots in Fig. 5.2. These plots show a 2D slice of the 4D state-action space. The top row shows the ground truth from simulation and the bottom row shows the learned approximator. We can see that the model performs well for most of the space, with the highest discrepancies occurring at the most extreme edges of the space. The failure margin function approximator in Fig. 5.3 captures the general shape of the function but does not replicate the zero contour extremely accurately. The ground truth plots show artifacts of our kd-tree method. The bubble-like arcs in the contour lines are a result of individual points in the kd-tree. This problem would get worse as dimensionality increases, but could be countered with more sophisticated sampling techniques to add points to the kd-tree that are near the success/failure boundary.

The most useful feature of the margin function is accurate classification of state-action points. We expect a perfect margin function to assign every valid point a positive value and every failure a negative value. To evaluate the performance, we can look at the classification accuracy as we vary our threshold, ϵ. This helps us choose the threshold for our optimization to be reasonably confident we will not accidentally allow failure states because of the inaccuracy of our approximation. However, as we increase the constraint threshold we will exclude more valid points which limits the space of possible behaviors. Fig. 5.4 shows the accuracy of the valid label and the percentage of valid points included as we vary the threshold, ϵ. From this, we make the engineering decision to use a threshold of 0.05. This will result in 97% of valid labeled points being valid and 59% of possible valid points being included.

5.6.2 Footstep Optimization and Failure Margin Utility

To test the utility of the approximations we implemented the multi step locomotion optimization problem from Section 5.5. We assigned our system a random initial height and forward speed in the valid range from (5.15), then sought a solution that over the course of N steps would achieve an apex state that matched a randomly selected goal height and forward speed. This approach did not command the final position of the robot because it was intended to emulate a gait transition rather than a position-keeping task. We implemented the nonlinear optimization problem using the cyipopt python wrapper around ipopt. [158]. The objective function gradient and constraint Jacobians were analytically calculated using pytorch's automatic differentiation functionality.

This optimization problem is simple enough that we can form a strong intuition for the optimal motion and when failures could manifest. The objective function from (5.9) is minimized by having a linearly varying apex height and forward velocity along the planned motion. If it is physically possible, the optimal motion will be to select whatever leg angle and extension results in these linearly varying apex states. however if this motion is not possible or allowed by the margin constraint, the optimizer should select the closest to linearly varying apex states as is allowable. Generally, we expect this problem will fail when requiring large

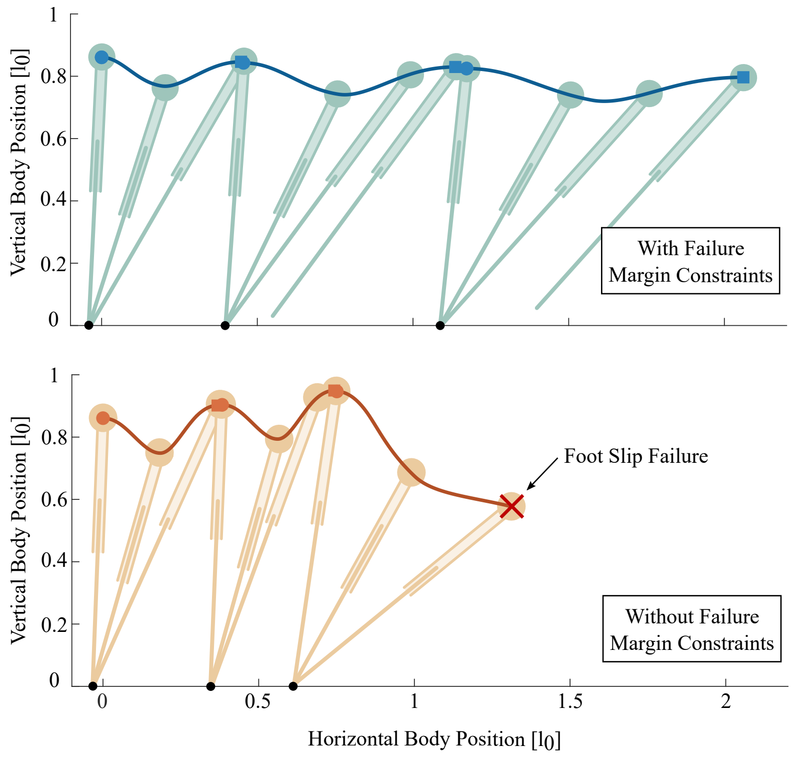


Figure 5.5: Resulting solutions from optimization problems where the task to significantly speed up and slightly lower the apex height. The optimization problem with failure margin constraints produce a valid trajectory and the problem without failure margin constraints solves “successfully” but contains a failure step.

changes in velocity in a short number of steps, because that would be the instance of extreme leg angles leading to foot slip.

A difficult planning task is illustrated in Fig. 5.5. In this figure we show the resulting solution of the optimization problem with and without margin function constraints. Additionally we simulated each commanded step and plotted the motion of the body through space. The first return map approximation is not perfect so we see small defects between the end of the simulated motion and the optimized intermediate apex state. The optimization without failure margin constraints thinks it found a valid plan, but we can see that step three actually results in foot slip as it falls forward.

Problem Description	3 Step Horizon			
	State Objective		No Objective	
	Margin	No Margin	Margin	No Margin
Failed to Solve	25.4 %	1.1 %	29.0 %	0.5 %
Contains Failure Step	27.6 %	56.5 %	13.0 %	60.0 %
Valid Solution	47.0 %	42.4 %	57.6 %	39.5 %
Mean Time (sec)	0.29	0.18	0.14	0.072

Problem Description	4 Step Horizon			
	State Objective		No Objective	
	Margin	No Margin	Margin	No Margin
Failed to Solve	29.4 %	10.5 %	22.6 %	0.4 %
Contains Failure Step	27.8 %	51.9 %	11.9 %	56.1 %
Valid Solution	42.8 %	37.6 %	65.5 %	43.5 %
Mean Time (sec)	1.29	0.98	0.16	0.079

Table 5.1: Performance of variations on the motion planning problem across 1000 random tasks.

To examine if the failure margin function improves the reliability of the optimization problem, we ran 1000 motion planning problems with three and four step planning horizons. The results of these optimizations are summarized in Table 5.1. In every situation, the failure margin constraints increase the percentage of problems for which a fully valid solution was found , with a relative increase of between 11% (three step, with state objective) and 51% (four step, without state objective). The optimizations with failure margin function constraints returns no solution more often than those without, but they have a greatly reduced chance of returning a solution that contain failure steps. Similar to previous studies, we see that the problems without objectives solve more reliably in significantly less time, particularly for the longer horizon case.

5.7 Conclusions

The approach we presented in this paper shows that it is possible to efficiently plan robot locomotion multiple footsteps ahead with a complex model that can be designed for high quality locomotion, not mathematical ease. In particular, our failure margin function greatly improved performance for models that exhibit catastrophic failure conditions. This approach was demonstrated on a canonical model of locomotion that is well understood but does not have a closed form solution. The failure margin function constraints were able to increase the optimization's ability to produce a valid plan by between 11 and 51 percent. Our proof of concept implementation is on the edge of speeds necessary for real-time planning. Further engineering effort in optimizing the approximator sizes, tuning optimizer tolerances and optimizing the software implementation could provide a meaningful decrease in execution time.

While we demonstrated this problem on a canonical model of locomotion, there are no inherent barriers to applying this to real, full-dimensional robots. This planning method hold promise for application to a more physically grounded model such as the data-driven, reduced order models proposed in [28]. The complex dynamics that such models exhibit can be encapsulated cleanly into the controlled first-return map. An alternative approach is to start with a stabilizing locomotion controller for a full order robot, such as in [142, 57, 155], and sampling a Poincarè section of the full robot state and controller commands. The dimensionality of this return map is too large to plan with, but we could build an autoencoder to compress the states to a reduced representation. This would allow us to plan valid motions and control policy commands in this compressed state-action space.

Acknowledgements

We would like to thank Helei Duan, Jeremy Dao, Connor Yates, Aseem Saxena and Prof. Alan Fern for their thoughtful discussions and feedback on this work.

Part III

Learning Reactive Control from First Principles of Legged Locomotion

Chapter 6: Incentivizing Robust Gaits using Environment Distributions in Reinforcement Learning

This chapter contains a study which proposes an approach to learn control for traversing stairs without perception or world knowledge. It accomplishes this through careful design of the distribution of environments seen by the system in training together with the power of recurrent neural networks. We further investigate the strategies used by the learned policy and identify biologically recognizable gait features.

Contributions

Jonah Siekmann devised and implemented the learning approach, conducted hardware experiments and wrote the manuscript. Kevin Green performed the behavior analysis, contributed to the learning approach, assisted hardware experiments and co-wrote the manuscript. John Warilla contributed to the learning implementation, assisted hardware experiments and co-wrote the manuscript. Alan Fern and Jonathan Hurst supervised the investigation and co-wrote the manuscript.

Blind Bipedal Stair Traversal via Sim-to-Real Reinforcement Learning

Jonah Siekmann, Kevin Green, John Warila, Alan Fern, and Jonathan Hurst

Robotics: Science and Systems XVII
Virtual
July 2021

6.1 Abstract

Accurate and precise terrain estimation is a difficult problem for robot locomotion in real-world environments. Thus, it is useful to have systems that do not depend on accurate estimation to the point of fragility. In this paper, we explore the limits of such an approach by investigating the problem of traversing stair-like terrain without any external perception or terrain models on a bipedal robot. For such blind bipedal platforms, the problem appears difficult (even for humans) due to the surprise elevation changes. Our main contribution is to show that sim-to-real reinforcement learning (RL) can achieve robust locomotion over stair-like terrain on the bipedal robot Cassie using only proprioceptive feedback. Importantly, this only requires modifying an existing flat-terrain training RL framework to include stair-like terrain randomization, without any changes in reward function. To our knowledge, this is the first controller for a bipedal, human-scale robot capable of reliably traversing a variety of real-world stairs and other stair-like disturbances using only proprioception.

6.2 Introduction

In order to be useful in the real world, bipedal and humanoid robots need to be able to climb and descend stairs and stair-like terrain, such as raised platforms or sudden vertical drops, which are common features of human-centric environments. The ability to robustly navigate these environments is crucial to getting robots to work with and alongside humans safely. Achieving this level of robustness on a bipedal platform is no easy task; while other platforms such as quadrupedal robots benefit from inherent stability due to multiple points of contact with the ground at a given time and the ability to stop and stand like a table, bipedal robots such as Cassie rely entirely on dynamic stability (essentially always existing in a state of falling). On stair-like environments, this is especially apparent due to the difficulty of recovery from missteps with only two legs.

By contrast, robots with quadrupedal morphologies have been able to use pro-

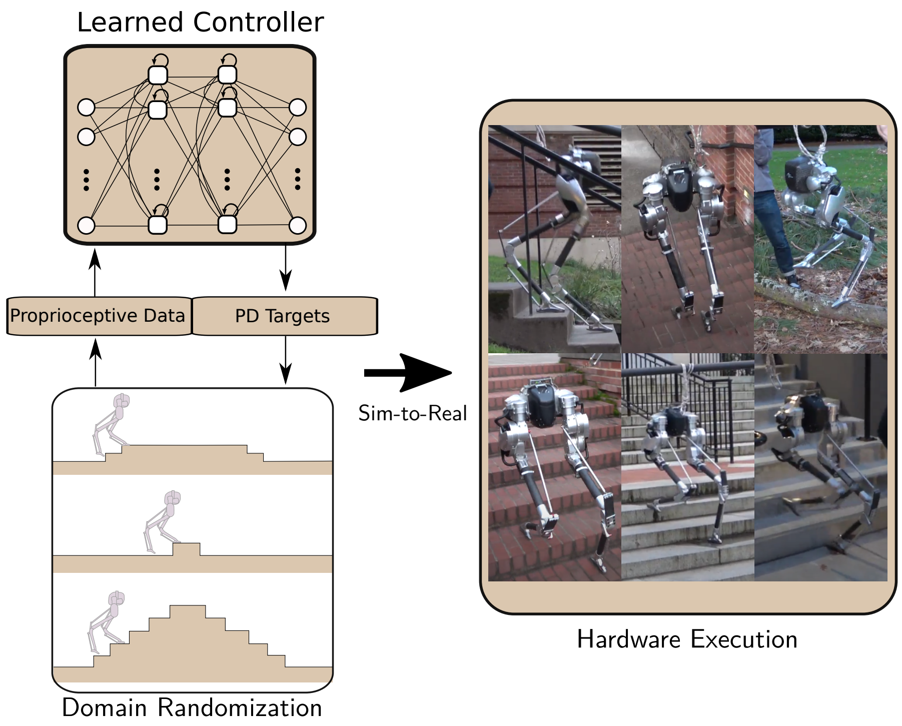


Figure 6.1: In this work, we investigate the limits of blind bipedal locomotion. We present a training pipeline which produces policies capable of blindly ascending and descending stairs in the real world. These policies learn proprioceptive reflexes to reject significant disturbances in ground height, resulting in highly robust behavior to many real-world environments.

prioception alone to negotiate stairs [96, 18], and hexapedal robots have even been able to use open-loop control to ascend and descend stairs [107]. While planar bipedal robots have been shown to be able to reject disturbances like large unexpected dropsteps [109], the vast majority of approaches seeking to enable such robots to negotiate stairs in the real world require either accurate vision systems [61, 6, 106] or operation in a carefully controlled laboratory environment [26, 132, 53], meaning the robot is localized through a known start location or the stairs are designed in tandem with robot morphology.

However, robots must be able to operate outside of controlled laboratory conditions and handle the massive variety of conditions in the real world. This goal is not compatible with a complete reliance on exteroceptive sensors such as RGB and depth cameras for accurate terrain estimation, which introduce fragility to real world conditions [54]. For instance, cameras may be unreliable if exposed to occlusion, fog, or varying lighting conditions. Further, integrating a state-of-the-art computer vision system into a high-speed controller is technically difficult, especially on a computationally limited platform like a mobile robot. For practical purposes, underlying controllers should be as robust as possible while relying on as little information about the world as possible. Ideally, a bipedal robot should be able to traverse as much of the entire breadth of human environments as possible using proprioception, while relying on exteroceptive sensing for further efficiency and high-level planning (and being robust to mistaken perception). This begs the question: how robust can a blind bipedal robot be?

Reinforcement learning (RL) based approaches have begun to show significant promise at robust real-world legged locomotion [96, 161, 141]. Unlike optimization or heuristic-based control methods which rely on prescribed ground contact schedules or force-based event detection, RL can produce control policies which learn proprioceptive reflexes and strategies for dealing with unexpectedly early or late contact and rough terrain through exposure to a variety of disturbances during training. However, the limits of this approach are unclear and prior work has not been demonstrated on the scale and variety of disturbances involved in stair-like terrain.

In this work, we show that robust proprioceptive bipedal control for complex stair-like terrain can be learned via an existing RL framework with surprisingly little modification. In particular, the only adjustment needed is the terrain randomization used during training, where we define a distribution over upward and downward going stairs including variation in height, width, and slope of the contact planes. Learning on this distribution allows for blind locomotion up and down unknown stairs as well as handling more general stair-like terrain characteristics, e.g. logs, curbs, dropoffs, etc. The learned controller is demonstrated in simulation and a variety of real-world settings. To our knowledge, this is the first demonstration of its kind and suggests the continued exploration into the limits of robust proprioceptive bipedal control.

6.3 Reinforcement Learning Formulation

We follow a sim-to-real reinforcement learning (RL) approach for learning bipedal locomotion and assume basic familiarity with RL [148]. In the general RL setting, at each discrete time step t the robot control policy π receives the current state s_t and returns an action a_t, which is applied and results in a transition to the next state s_{t+1}. The state transition dynamics are unknown to the robot and are governed by a combination of environmental conditions, such as terrain type, and the robot dynamics. In addition, during learning, each state transition is associated with a real-valued reward r_t. The reward is governed by the application goals to encourage the desired behavior during learning. The RL optimization objective considered in this work is to learn a policy through interaction with the environment that maximizes the expected cumulative discounted reward over a finite-horizon T. That is, find a policy π that maximizes: $J(\pi) = \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t R_t\right]$, where $\gamma \in [0, 1]$ is the discount factor and R_t is a random variable representing the reward at time t when following π from a state drawn from an initial state distribution.

For complex environments, RL typically requires large amounts of training experience to identify a good policy. Further, for biped locomotion, the training will involve many falls and crashes, especially early in training. Thus, training from scratch in the real-world is not practical and we instead follow a sim-to-real RL paradigm. Training is done completely in a simulation environment, with dynamics randomization (see below), and the resulting policy is then used in the real-world.

In the remainder of this section, we detail the specific sim-to-real RL formulation used in this work, which follows recent work [141] on learning different biped gaits over flat terrain. Surprisingly, only minimal changes were required to en-

Command	Probability of Change	Range
Forward Speed	1/300	[-0.3m/s, 1.5m/s]
Sideways Speed	1/300	[-0.3m/s, 0.3m/s
Turn Rate	1/300	[-90deg/s, 90deg/s]

Table 6.1: At each timestep, each command input to the policy is subject to a 1/300 probability of being altered. When this occurs, a new command is sampled from a uniform distribution parameterized by the rightmost column. Given that maximum episode length is 300 discrete timesteps, this means each command will change at least once on average per episode.

able policy learning for the much more complex stair-like terrains of this paper[1] In particular, the only major modification required was the randomized domain generation of stair-like rather than mostly flat terrain as discussed later in Section 6.4; no novel stair-specific reward terms were needed.

6.3.1 State Space

The state s_t that is input to the control policy at each time step includes three main components. First, the state contains information about the robot's instantaneous physical state, including the pelvis orientation in quaternion format, the angular velocity of the pelvis, the joint positions, and the joint velocities. The second component of s_t is composed of command inputs, which come from a human operator. These commands are subject to randomization during training to give the policies a wide breadth of experience attempting to traverse stairs over a variety of speeds and approach angles. Details of this randomization can be seen in Table 6.1.

The third component includes two cyclic clock inputs, each corresponding to a

[1]This was only discovered after a careful ablation analysis of our first success on stair-like terrain, which originally included seemingly necessary modifications to prior work, such as more complex reward functions and state features.

leg of the robot, p:

$$p = \begin{cases} \sin\left(2\pi(\phi_t + 0.0)\right) \\ \sin\left(2\pi(\phi_t + 0.5)\right) \end{cases} \tag{6.1}$$

Here ϕ_t is a phase variable which increments from 0 to 1, then rolls back over to 0, keeping track of the current phase of the gait. The constant offsets 0.0 and 0.5 are phase offsets used to make sure that the left and right legs are always diametrically opposite of each other in terms of phase during locomotion.

6.3.2 Action Space

The output action a_t of the control policy at each time step (running at 40Hz) is an 11 dimensional vector with the first 10 entries corresponding to PD targets for the joints, each of which are fed into a PD controller for each joint operating at 2KHz. Prior work has found it advantageous to learn actions in the PD target space rather than directly learning the higher-rate actuation commands [113].

The final dimension of a_t is a clock delta δ_t (refer to 6.3.1 for information on clocks), which allows the policy to regulate the stepping frequency of the gait. Intuitively, this allows the controller to choose an appropriate stepping frequency for a particular gait, command, and terrain. Specifically, the phase variable ϕ in the state representation (Section 6.3.1) is updated at each timestep t by,

$$\phi_{t+1} = \text{fmod}(\phi_t + \delta_t, 1.0). \tag{6.2}$$

This delta is bounded in a way such that the policy can choose to regulate the gait cycle between 0.5x and 1.5x the nominal stepping frequency (which is approximately one gait cycle every 0.7 seconds). While this component is included in the control policy action, it does not appear to have a large impact on performance and the learned policy does not vary δ_t much in response to disturbances. We suspect that future ablation analysis will show that it is not important for performance on the real robot.[2]

[2]We leave this as a hypothesis here, since we have not been able to test on the real robot at

6.3.3 Reward Function

To briefly review the approach taken in [141], we wish to take advantage of the complementary nature of foot forces and foot velocities during locomotion to construct a reward function which will punish one and allow the other, and vice versa, at key intervals during the gait. We use a probabilistic framework to represent uncertainty around the timings of these intervals. More specifically, we make use of a binary-valued random indicator function $I_i(\phi)$ for each quantity q_i which we wish to penalize at some time during the gait cycle. This indicator function is likely to be 1 during the interval in which it is active, and likely to be 0 during intervals in which it is not active. The distribution of this binary-valued random function is defined via the Von Mises distribution; for a more comprehensive description, see [143]. In addition, rather than use the actual random variable in the reward we instead opt to use its expectation for more stable learning; see Fig. 6.2 for a plot of this expectation.

Our full reward function is as follows;

$$R(s, \phi) = 1 - \mathbb{E}[\rho(s, \phi)] \tag{6.3}$$

Which is to say, our reward is the difference of a bias and the expectation of a probabilistic penalty term $\rho(s, \phi)$ as described in [141]. See Table 6.2 for detailed information on the exact quantities and weightings used.

We define F_l and F_r as the vectors of translational forces applied to the left and right foot, and v_l and v_r similarly as the vectors of left and right foot velocities. To maintain a steady orientation, an orientation error ϵ_o is used, which is equal to,

$$\epsilon_o = 3(1 - \hat{q}^T q_{\text{body}})^2 + 10\left((1 - \hat{q}^T q_{\text{l}})^2 + (1 - \hat{q}^T q_{\text{r}})^2\right) \tag{6.4}$$

where q_{l} and q_{r} are the quaternion orientations of the left and right foot, q_{body} is the quaternion orientation of the pelvis, and $\hat{q}$ is a desired orientation (for our purposes, fixed to be always be facing straight ahead).

the time of submission.

Weight	Cost Component
0.140	$1 - \mathbb{E}[I_{\text{left force}}(\phi)] \cdot \exp(-.01\|F_l\|)$
0.140	$1 - \mathbb{E}[I_{\text{right force}}(\phi)] \cdot \exp(-.01\|F_r\|)$
0.140	$1 - \mathbb{E}[I_{\text{left velocity}}(\phi)] \cdot \exp(-\|v_l\|)$
0.140	$1 - \mathbb{E}[I_{\text{right velocity}}(\phi)] \cdot \exp(-\|v_r\|)$
0.140	$1 - \exp(-\epsilon_o)$
0.140	$1 - \exp(-\lvert\dot{x}_{\text{desired}} - \dot{x}_{\text{actual}}\rvert)$
0.078	$1 - \exp(-\lvert\dot{y}_{\text{desired}} - \dot{y}_{\text{actual}}\rvert)$
0.028	$1 - \exp(-5 \cdot \|a_t - a_{t-1}\|)$
0.028	$1 - \exp(-0.05 \cdot \|\tau\|)$
0.028	$1 - \exp(-0.1(\|\text{pelvis}_{\text{rot}}\| + \|\text{pelvis}_{\text{acc}}\|))$

Table 6.2: The cost terms which are summed together to compose the expected penalty, $\mathbb{E}[\rho(s, \phi)]$. Terms involving an expectation of a variable $I_i(\phi)$ vary over the course of the gait cycle, with the goal of penalizing foot forces and foot velocities at key intervals to teach the policy to lift and place the feet periodically in order to walk. Other terms exist for the sake of commanding the policy to move forward, backwards, or sideways, or turn the robot to face a desired heading. Finally, the remaining terms exist to reduce behaviors which are shaky and thus unlikely to work well on hardware.

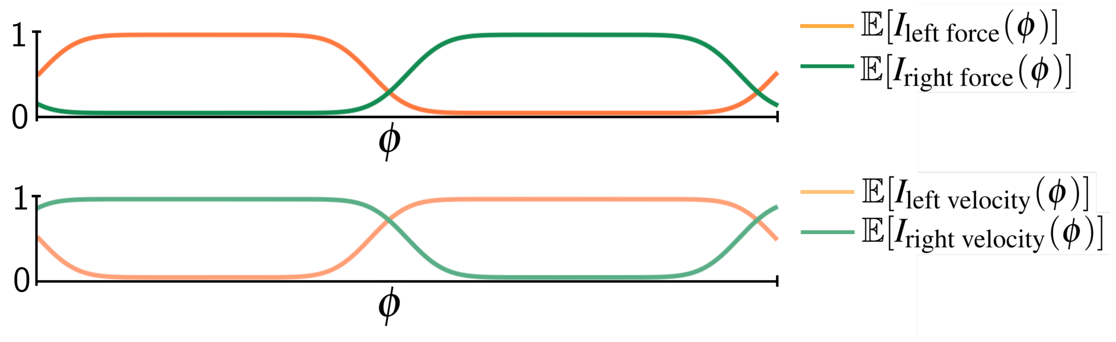


Figure 6.2: By alternatingly punishing foot forces during a 'stance' phase to teach the policy to lift the foot, and punishing foot velocities during a 'swing' phase to teach the policy to place the foot on the ground, we can construct a foundation on which to learn simple walking behavior. Following in the path of previous work, we define these cyclic coefficents as random indicator functions of the phase, and take their expectation.

The quantities $\dot{x}_{\text{desired}}$ and $\dot{y}_{\text{desired}}$ correspond to a commanded translational speed, while $\dot{x}_{\text{actual}}$ and $\dot{y}_{\text{actual}}$ are the actual translational speed of the robot. The term $\text{pelvis}_{\text{rot}}$ represents the angular velocity while $\text{pelvis}_{\text{acc}}$ represents translational acceleration; these terms are used in the cost component to reduce the shakiness of locomotion behavior. The terms a_t and a_{t-1} refer to the current timestep's action and the previous timestep's action, and their use in the cost component is to encourage smooth behaviors. The term τ is the vector of net torques applied to the joints, and its use in the cost component is intended to encourage energy efficient gaits.

6.3.4 Dynamics Randomization

In order to overcome any modeling errors that may be present in our simulated Cassie environment, we randomize several important quantities of the dynamics at the beginning of each episode during training as in previous work [112] [141]. These randomized parameters are listed in Table 6.3.

Parameter	Unit	Range
Joint damping	Nms/rad	$[0.5, 3.5] \times$ default values
Joint mass	kg	$[0.5, 1.7] \times$ default values
Ground Friction	–	$[0.5, 1.1]$
Joint Encoder Offset	rad	$[-0.05, 0.05]$
Execution Rate	Hz	$[37, 42]$

Table 6.3: To prevent overfitting to simulation dynamics and facilitate a smooth sim-to-real transfer, we employ dynamics randomization. The above ranges parameterize a uniform distribution for each listed parameters. Damping, mass, friction, and encoder offset are randomized at the beginning of each rollout, while execution rate is randomized at each timestep to mimic the effect of variable system delay on the real robot.

6.3.5 Policy Representation and Learning

We represent the control policy as an LSTM recurrent neural network [70], with two recurrent hidden layers of dimension 128 each. We opt to use a memory-enabled network because of previous work demonstrating a higher degree of proficiency in handling partially observable environments [67] [112] [143]. For ablation experiments, involving non-memory-based control policies, we use a standard feedforward neural network with two layers of dimension 300, with tanh activation functions, such that the number of parameters is approximately equal to that of the LSTM network.

For sim-to-real training of the policy, we use Proximal Policy Optimization (PPO) [136], a model-free deep RL algorithm. Specifically, we use a KL-threshold-termination variant, wherein each time the policy is updated, the KL divergence between the updated policy and the former policy is calculated and the update is aborted if the divergence is too large. During training, we make use of a mirror loss term [3] in order to ensure that the control policy does not learn asymmetric gaits. For recurrent policies, we sample batches of episodes from a replay buffer as in [143], while for feedforward policies we sample batches of timesteps. Each episode is limited to be 300 timesteps, which corresponds to about 7.5 seconds of

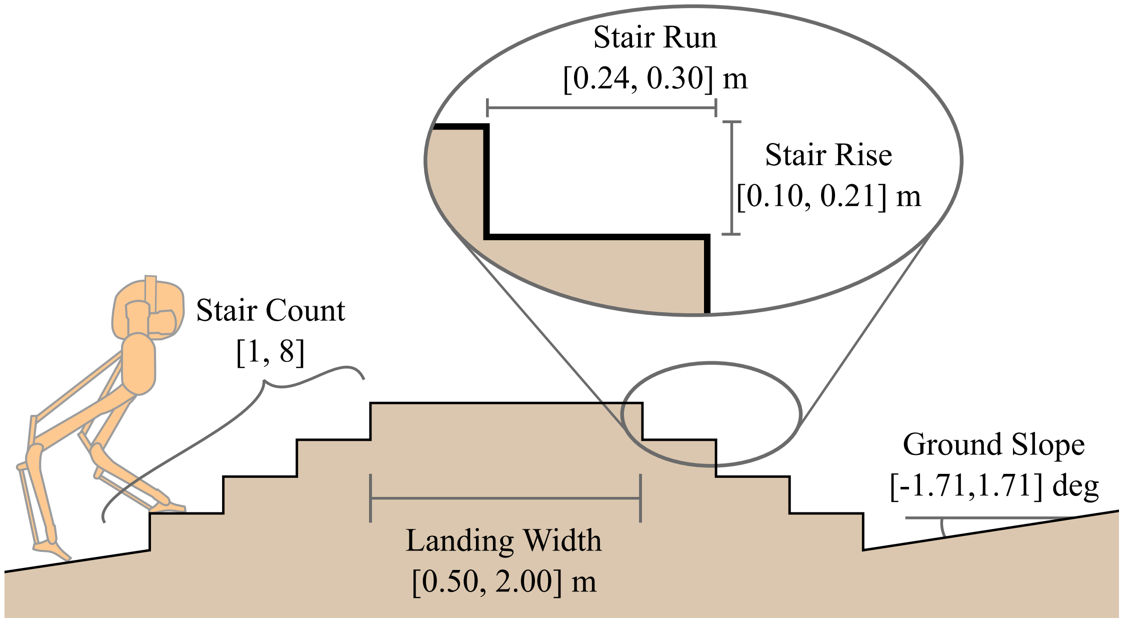


Figure 6.3: In order to ensure robustness over a variety of possible stair-like terrain, we randomize a number of parameters when generating stairs at the start of each episode in simulation. These parameters include the number of stairs, the height of each stair, the length of each stair, the length of the landing atop the stairs, and the slope of the ground immediately before and after the stairs.

simulation time.

6.4 Terrain Randomization

Previous work on applying RL to Cassie has either trained on flat ground [161] [143] or on randomized slight inclines [141]. Other work in applying deep RL has investigated employing a curriculum of rough terrains which become increasingly difficult as training progresses [96]. For the purpose of simplicity, we find that training on interactions with a randomized staircase without a curriculum is sufficient to learn robust behavior.

To this end, we train on a plane whose incline is randomized at the beginning of each rollout in the pitch and roll axes. This incline is between -0.03 radians and 0.03

radians. As part of the dynamics randomization, ground friction is randomized, increasing the potential difficulty of the environment. The starting position of the stairs are randomized at the beginning of each rollout, such that the episode can start with the policy already on top of the stairs, or with the stairs up to 10 meters in front of the policy. This is done in order to ensure that the policy is able to see lots of experience on flat or inclined ground, as well as on stairs.

The dimensions of the stairs are randomized within typical city code dimensions, with a per-step rise of between 10cm and 21cm, and a run of 24cm to 30cm. The number of stairs is also randomized, such that each set of stairs has between 1 and 8 individual steps. A small amount of noise ($\pm$ 1cm) is added to the rise and run of each step such that the stairs are never entirely uniform, to prevent the policy from deducing the precise dimensions of the stairs via proprioception and subsequently overfitting to perfectly uniform stairs.

6.5 Results

We trained four groups of policies, each containing five policies initialized with different random seeds. First, we trained a group of simple LSTM policies with stair terrain randomization; these are referred to in this section as **Stair LSTM**. To investigate the importance of memory, we trained a group of feedforward policies also with stair terrain randomization; we denote these **Stair FF**. We also trained a group of policies without stair terrain randomization, and denote these **Flat Ground LSTM**, to investigate the importance of the terrain randomization introduced in this work. The final group was trained with a simple additional binary input informing the policy whether or not stairs were present within one meter of the policy, referred to here as **Proximity LSTM**, in order to investigate the benefit of leaking information about the world to the policies.

Each policy was trained until 300 million timesteps were sampled from the virtual environment, simulated with MuJoCo [152]. Our selection of hyperparameters for the PPO algorithm includes a replay buffer size of 50,000 timesteps, a batch size of 64 trajectories for recurrent policies, and a batch size of 1024 timesteps

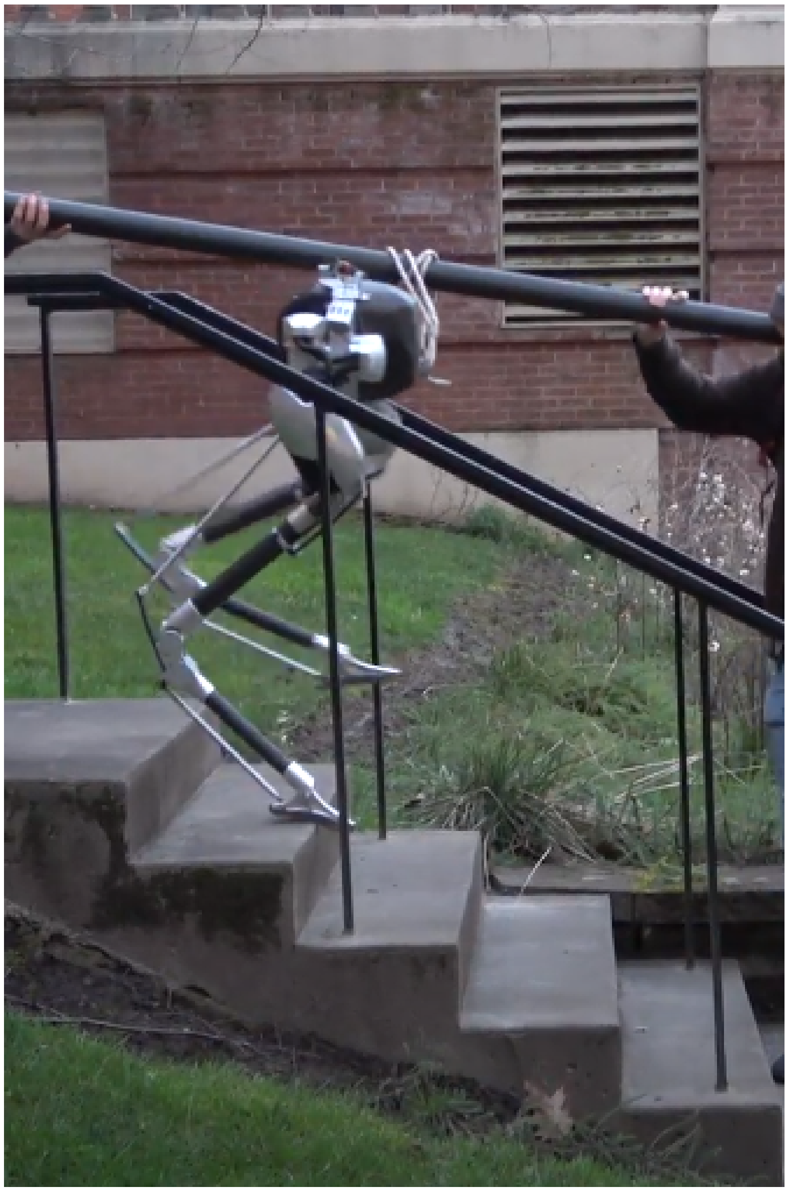


Figure 6.4: The learned policies exhibit a high degree of blind robustness to a variety of stair-like terrain, and can reliably ascend and descend stairs of typical dimensions found in human environments.

for feedforward policies. Each replay buffer is sampled for up to five epochs, with optimization terminating early if the KL divergence reached the maximum allowed threshold of 0.02. We clear our replay buffer at the start of each iteration. We use the Adam [92] optimizer with a learning rate of 0.0005 for both the actor and critic, which are learned separately and do not share parameters.

6.5.1 Simulation

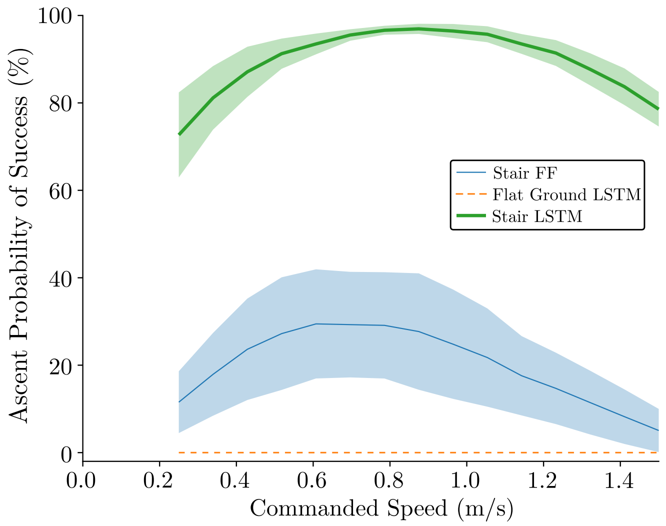
6.5.1.1 Probability of Successfully Ascending and Descending Stairs

To understand the importance of memory and terrain randomization, we evaluate three groups of policies on the task of successfully climbing and descending a set of stairs in simulation. We compare the performance of Stair FF, Stair LSTM, and Flat Ground LSTM policies on this task.

Specifically, we run 150 trials testing how often a policy is successfully able to climb a set of stairs with five steps, each with a tread of 17cm and a depth of 30cm (a typical real-world and relatively mild stair geometry). This should give us an estimate of how reliably each group of control policies can climb a flight of stairs that it approaches blindly. Success is defined as reaching the top of the flight of stairs without falling. We also apply this procedure for descending stairs, running 150 trials on stairs with the same dimensions, and record the rate at which each group of policies can reach the bottom without falling.

The results of these tests for three different training conditions is shown in figure 6.5. We note that the Stair LSTM policy has the highest overall probability of success. Nevertheless, the probability of success is dependent, in large part, on approach speed. The policies experience higher rates of failure at low speeds, where they may lack the momentum to propel themselves past poorly chosen foot placements. They also experience higher rates of failure at high speeds, possibly due to the more dynamic nature of high-speed gaits.

The Flat Ground LSTM policies, having never seen stair-like terrain during



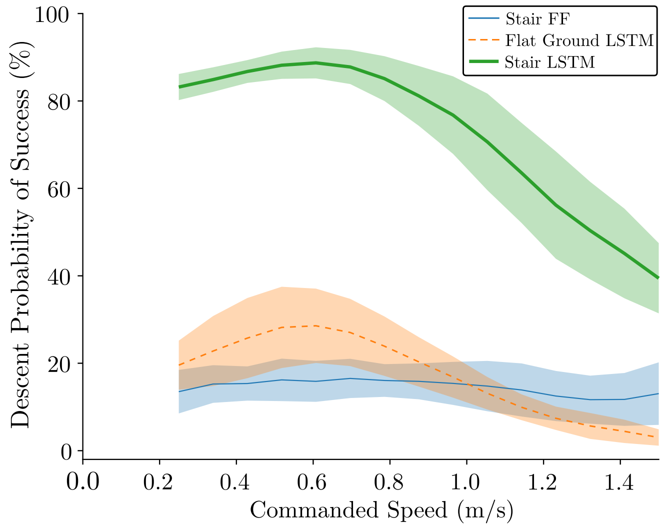


Figure 6.5: We evaluate the probability of successfully climbing and descending stairs without falling as a function of commanded speed between 0.25 m/s and 1.5 m/s over 150 trials. For Stair LSTM policies, there seems to be an optimal approach speed for climbing stairs and a separate optimal descent speed. Stair FF policies do not attain high performance, implying that memory could be an important component of the learned behavior. Flat Ground LSTM policies, having never encountered stairs in training, are virtually unable to climb stairs while finding some success in descending stairs without falling over.

training, are unable to compensate and experiences a high rate of failure for both ascent and descent. The Stair FF policies, despite encountering stairs during training, are unable to learn an effective strategy for handling stairs, implying that memory may be an important mechanism for robustness to stair-like terrain.

6.5.1.2 Energy Efficiency Comparison

To understand the consequences of training with terrain randomization, we also compare the cost of transport between Flat Ground LSTM policies, Stair LSTM policies, and Proximity LSTM policies. The cost of transport (CoT) is a common measure of efficiency of legged robots, humans and animals. It is the energy used per distance, normalized by weight to be unitless. It is defined as

$$\text{CoT} = \frac{E_\text{m}}{Mgd}, \tag{6.5}$$

where E_m is the energy used by the motors, M is the total mass of the robot, g is the gravitational acceleration and d is the distance traveled. The energy used by Cassie is calculated using positive actuator work and resistive losses via

$$E_\text{m} = \int_0^T \Big(\sum_i max(\tau_i \cdot \omega_i,\ 0) + \frac{\omega_i^\text{max}}{P_i^\text{max}}\tau_i^2\Big)dt. \tag{6.6}$$

Here τ_i is the torque applied to motor i and ω_i is its rotational velocity. We use two parameters to define the resistive losses in terms of torque, P_i^{max} is the maximum input power and ω_i^{max} is the maximum speed of motor i. The results of testing steady state CoT at 1 m/s on flat ground can be seen in Table 6.4. These calculations of CoT do not include the overhead power draw from computation and control electronics so they should not be used to compare between robots, only between control policies.

We find that Flat Ground LSTM policies learn the most energy efficient gaits for walking on flat ground. Stair LSTM policies learn less efficient flat-ground gaits

Policy Group	Mean CoT	Std. CoT
Proximity LSTM (stairs)	0.47	0.0086
Stair LSTM	0.46	0.0323
Proximity LSTM (flat)	0.39	0.0257
Flat Ground LSTM	0.38	0.0205

Table 6.4: Locomotion efficiency as measured by cost of transport (CoT) for walking at 1 m/s over flat ground in simulation between three groups of policies over all five random seeds. We note that policies not trained on stair terrain randomization tend to learn more energy efficient gaits, though some energy efficiency can be recovered by providing the stair-trained policies with a binary stair presence/absence input.

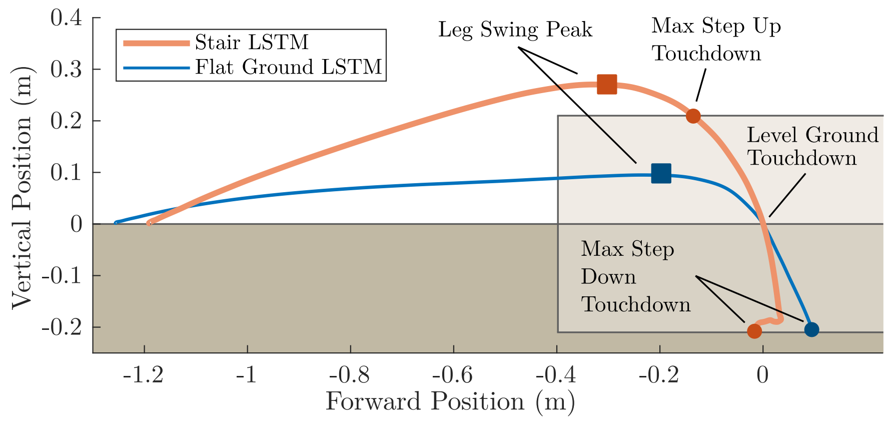
in order to be robust to stairs; however, the stair proximity input to the Proximity LSTM can help to recover some of this lost energy efficiency by allowing the learned controller to switch between a stair-ready gait and a more energy efficient, flat-ground gait.

6.5.2 Behavior Analysis

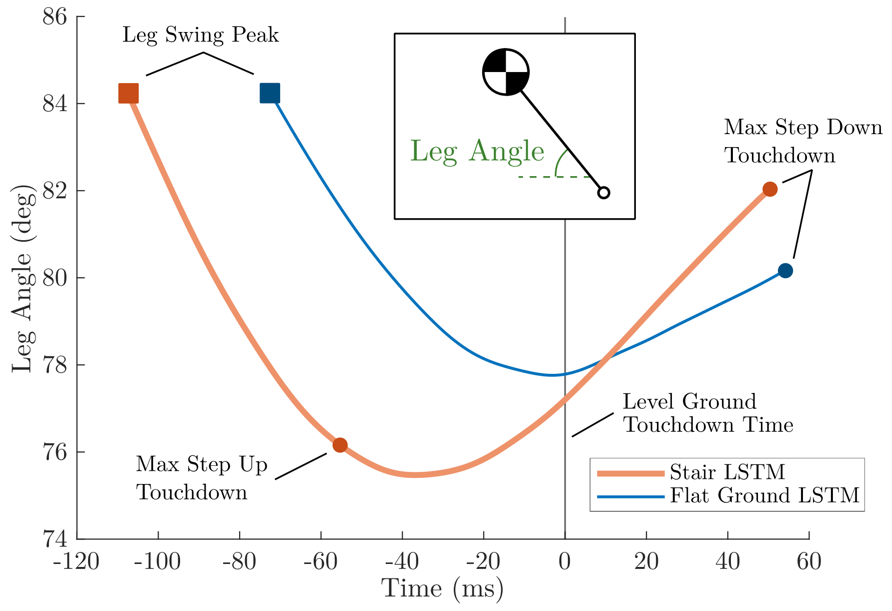
To understand the strategy adopted by the policy, we can benefit from taking the perspective of experimental biology.We specifically look at the behavior as the robot contacts the first step up or down after walking along flat ground. First we will analyze the swing leg motion to understand how the robot places its foot on step ups and step downs. Once the swing foot contacts a step up or down, the force applied by the foot on the ground during stance phase can be modulated to better prepare the robot for future steps. We analyze how the ground reaction force and total impulse varies in the case of step ups and step downs.

6.5.2.1 Swing Foot Motion

To understand the change the stair terrain makes in the foot swing path we compare the result of a Flat Ground LSTM policy and a Stair LSTM policy when they



(a) Swing foot paths for the stair trained policy and the flat ground policy overlaid on example step ups and step downs.



(b) The leg angle between the robot body and the swing foot as the foot descends toward touchdown.

Figure 6.6: A comparison of the swing foot motion of the Stair LSTM policy and the Flat Ground LSTM policy while locomoting at 1.0 m/s. There is a significant change in the leg swing policy as a result of training on randomized stairs. The most significant changes are higher foot clearance, a steeper foot descent and a faster leg angle retraction rate.

encounter a drop step. The foot swing path during a drop step lets us see where the policy would place the foot if it had encountered a step up or a step down. Fig. 6.6a shows the foot swing path of these two policies relative to the ground. We can see that the Stair LSTM policy takes a much higher step compared to the Flat Ground LSTM policy which gives it additional clearance so it can step up onto a large step. A second interesting observation is the steeper path of the swing foot for the Stair LSTM policy. The swing foot only moves forward 14 cm while it is in the height range where it may encounter the front face of a step up. We hypothesize this is a strategy that prevents the foot from stubbing the toe too hard on the front face of a stair and causing the robot to trip forward.

A second viewpoint to understand leg swing motion is to look at the leg swing retraction. In humans and in bipedal birds it is observed that the swing leg is swung backwards, relative to the body, towards the ground near the end of stance [116, 39]. This has the benefits of reducing the velocity of the foot relative to the ground and thus reducing the impact [20] as well as improving ground height disturbance rejection by automatically varying the leg touchdown angle [139].

Our training procedure does not explicitly incentivize the policy to exhibit these leg swing retraction behaviors, but we do see them emerge as shown in Fig. 6.6b. This figure shows the angle of the swing foot relative to the body between the peak of leg swing and contact with the maximum step down. The Stair LSTM policy has a faster leg retraction rate compared to the Flat Ground LSTM policy. With only this data we cannot say if this retraction profile is optimal or even if it is the cause of the improved performance on stairs. However, the fact that there is a significant change in the leg retraction profile as a result of training on stairs is an interesting observation.

6.5.2.2 Ground Reaction Forces

Once the robot's foot has touched down its control authority is limited due to the underactuated nature of bipedal locomotion. However, the robot still has a significant amount of control through the ground reaction force. To understand

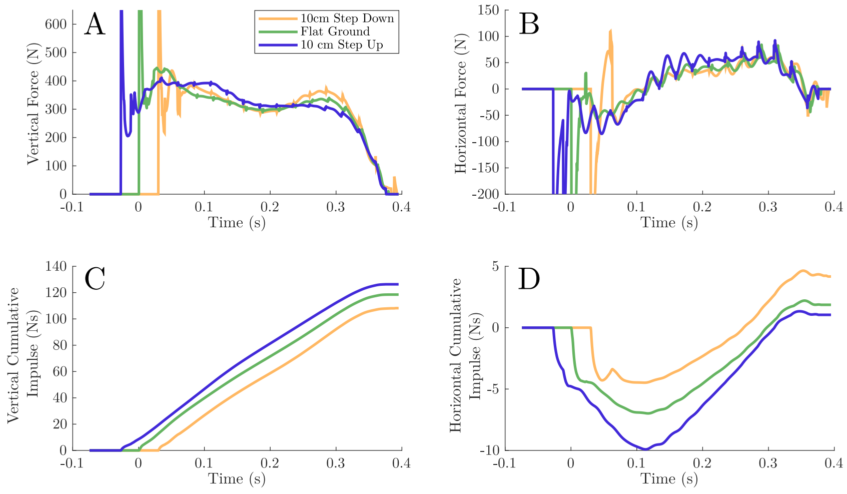


Figure 6.7: The ground reaction forces and cumulative impulses of a Stair LSTM policy when it encounters varying ground height. The peak vertical force (A) after the impact remain roughly equivalent while the force in the second half of stance is modulated. The horizontal force (B) shows oscillations that match the frequency of the learned policy execution rate. This may be the policy controlling the body's attitude. The total vertical impulse (C) shows the expected result of a larger impulse stepping up and a smaller one stepping down. The horizontal impulse (D) shows a result that is predicted by leg swing retraction. When stepping down the foot is shifted backwards relative to the body which results in net acceleration forward which is shown here by a positive horizontal impulse.

how the Stair LSTM policy reacts to a 10 cm step up or down we plot the horizontal and vertical ground reaction forces in the sagittal plane in Fig. 6.7. At the beginning of stance there is a large spike in force that dwarfs the normal forces during stance. The force value during this spike is largely defined by the tuning of the simulation contact model so it is not of primary interest to understanding the behavior of the policy. The first interesting thing we see in subplot A is that the maximum nominal leg force is held relatively constant which is a predicted result of a well adjusted leg swing policy [17]. Second we see that the magnitude of the second hump in the double humped ground reaction forces is increased in the step down and decreased in the step up. In the horizontal forces (subplot B) we see an oscillating signal where the oscillations match the frequency of policy evaluation. We hypothesize that this is the policy working to control the attitude of the pelvis. Prioritizing body attitude over forward velocity would be similar to the explicit priorities during single stance in Virtual Model Control [73]. The lower two subplots (C and D) show the cumulative impulse in the vertical and forward directions throughout the stance phase. We can see that the step up applies a larger vertical impulse and the step down a smaller vertical impulse. This agrees with the intuition that the robot should apply a smaller vertical impulse to lower itself down a step compared to lifting itself up a step. The horizontal impulse tells us if the robot speeds up or slows down in the forward direction during the stance phase. We see that the step down results in a significantly larger forward impulse and the step up reduces the vertical impulse very slightly. This aligns with the predicted behavior from a well tuned swing leg retraction policy.

6.5.3 Hardware

The recurrent policies transferred to hardware without any notable difficulties. We were able to take the robot for a walk around a large university campus using a randomly selected Stair LSTM policy and attempt to climb the staircases we came across. We observed robust and error-correcting behavior, as well as successful and repeatable stair ascents and descents. In addition, we noted robustness to uneven terrain, logs, and curbs, none of which were modeled in training. The policy was similarly robust to inclines and deformable terrain, demonstrated by a walk through a wet grass field and up a small hill. These experiments can be seen in our submission video [3], and a still image of one such experiment can be seen in Fig. 6.4.

In addition to testing one-off terrains all over the university campus, we ran ten trials ascending stairs, and ten trials descending stairs on an outdoor real-world staircase. We recorded an 80% success rate in ascending stairs using the selected Stair LSTM policy, and a 100% success rate in descending stairs. A full video of this trial can be seen in our attachment to this submission. We note that the learned behavior is robust to missteps, and can quickly recover from mistakes, though the policy is not completely infallible and will fall if it makes a particularly egregious error. This experiment can be seen in our supplemental video [4]. The blind, proprioceptive learned strategy appears to rely on a solid stair face; evaluating policies on slatted stairs in simulation resulted in a much higher failure rate, pointing to the limits of such an approach. Even when explicitly included in training, slatted stairs tended to trip up policies on ascent. By contrast, stairs with randomly inclined steps (e.g., ones where each step had a unique pitch and roll orientation) did not seem to be difficult for ascent or descent. Likewise, approaching and ascending stairs at an angle did not seem to be an issue for policies.

6.6 Conclusion

In this work, we have motivated the desirability of a highly robust but blind walking controller, and demonstrated that such a blind bipedal walking controller is capable of climbing a wide variety of real-world stairs. In addition, we note that producing such a controller requires very little modification to an existing training pipeline [141], and in particular no stair-specific reward terms; simply adding stairs to

[3][Web link to submission video: youtu.be/MPhEmC6b6XU]
[4][Web link to supplemental video: youtu.be/nuhHiKEtaZQ]

the environment with no further information is sufficient for learning stair-capable control policies. An important requirement of this learned ability appears to be a memory mechanism of some kind, probably due to the partially observable nature of the task of walking through unknown terrain while blind. In future work, it will be interesting to investigate how vision can be most effectively used to improve the efficiency and/or performance of a blind bipedal robot. Further, this work has demonstrated surprising capabilities for blind locomotion and leaves open the question of where the limits lie.

Acknowledgments

This work was partially supported by NSF grant IIS-1849343 and DARPA contract W911NF-16-1-0002. Special thanks to Intel for access to the vLab cluster, and to Yesh Godse, Jeremy Dao, and Helei Duan for advice and guidance.

Chapter 7: Guiding Learned Policies using Optimized Spring-Mass Models

This chapter contains a study which investigates the potential of training control policies for legged robots which follow motion commands from a reduced order model. In particular we look at the applicability of optimized motions of the bipedal actuated spring loaded inverted pendulum model. We show that the policy learned to replicate features of spring mass locomotion. This shows promise as a method of tracking motion plans from reactive planners built on simplified models.

Contributions

Kevin Green devised the concept, implemented the trajectory optimization and assisted in developing the learning procedure. Yesh Godse and Jeremy Dao lead the learning procedure and trained the controllers. Hardware experiments were conducted by Kevin Green, Yesh Godse and Jeremy Dao with help from other Dynamic Robotics Lab members. Alan Fern, Ross L. Hatton and Jonathan Hurst supervised the investigation. Kevin Green primarily wrote the manuscript and all other authors contributed to the writing and editing.

Learning Spring Mass Locomotion: Guiding Policies With a Reduced-Order Model

Kevin Green, Yesh Godse, Jeremy Dao
Ross L. Hatton, Alan Fern and Jonathan Hurst

IEEE Robotics and Automation Letters
Volume 6, Number 2, pages 3926-3932
March 17, 2021

7.1 Abstract

In this paper, we describe an approach to achieve dynamic legged locomotion on physical robots which combines existing methods for control with reinforcement learning. Specifically, our goal is a control hierarchy in which highest-level behaviors are planned through reduced-order models, which describe the fundamental physics of legged locomotion, and lower level controllers utilize a learned policy that can bridge the gap between the idealized, simple model and the complex, full order robot. The high-level planner can use a model of the environment and be task specific, while the low-level learned controller can execute a wide range of motions so that it applies to many different tasks. In this letter we describe this learned dynamic walking controller and show that a range of walking motions from reduced-order models can be used as the command and primary training signal for learned policies. The resulting policies do not attempt to naively track the motion (as a traditional trajectory tracking controller would) but instead balance immediate motion tracking with long term stability. The resulting controller is demonstrated on a human scale, unconstrained, untethered bipedal robot at speeds up to 1.2 m/s. This letter builds the foundation of a generic, dynamic learned walking controller that can be applied to many different tasks.

7.2 Introduction

a powerful approach to control agile and dynamic legged robots is to use a control hierarchy that combines specific domain knowledge of legged locomotion with the power of deep reinforcement learning. The long term goal is to enable robots to be able to navigate quickly through previously unseen environments with agility that approaches or exceeds that of humans and animals. The control hierarchy should consist of a low-level walking controller generated through reinforcement learning that can account for and exploit the passive dynamics of the physical robot. This low-level controller receives motion commands from a terrain-aware motion planner. The commands from the planner must be rich enough to sufficiently direct the walking controller while still being as simple as possible. This letter focuses on the learned *controller* and its interface, so we elect to use a library of precomputed motions (Fig. 7.1).

Learned controllers have incredible potential to create dynamic locomotion, but to be integrated into a control hierarchy we need an effective control interface. Dynamic legged locomotion is by its nature underactuated, hybrid, unstable, nonlinear, and must be able to operate with significant ground uncertainty. These challenges may be addressed by deep neural networks acting as controllers because of their ability to encode highly nonlinear control policies. However, if we would like to develop more complex behaviors such as autonomous navigation through unknown, obstacle filled environments, we will need to extend this approach. It may be possible to expand the learning problem so that the same policy that dynamically controls the robot also interprets the world around it and chooses how to move to the goal, but we choose not to take this approach because of the challenge of generalization to new tasks, sensors and environments. Instead, we seek to create a learned controller that can be directed by other intelligent systems in a modular hierarchy. Recent work explores the use of hierarchical learned control structures to quadrupedal locomotion. Some methods use both a high-level learned policy and a low-level learned policy [82, 154]. Another method combines a low-level model based controller with a high-level, learned gait planner [35].

Reduced-order models of locomotion capture the core dynamics of locomotion which make them a compelling control interface. The most common models used to plan motion for bipedal robots are inverted pendulum models, which consist of a point mass and massless legs that can apply forces through ground contact [85, 108, 11]. These models describe the underactuation of dynamic walking as well as the discrete choice of foot step locations while remaining simple enough to plan with in real time. The spring loaded inverted pendulum (SLIP) model is particularly applicable for agile locomotion because with simple, feed-forward policies it demonstrates strong stabilizing effects [59, 69, 78]. Many agile robots closely resemble the SLIP model [93, 121] or are designed with SLIP locomotion

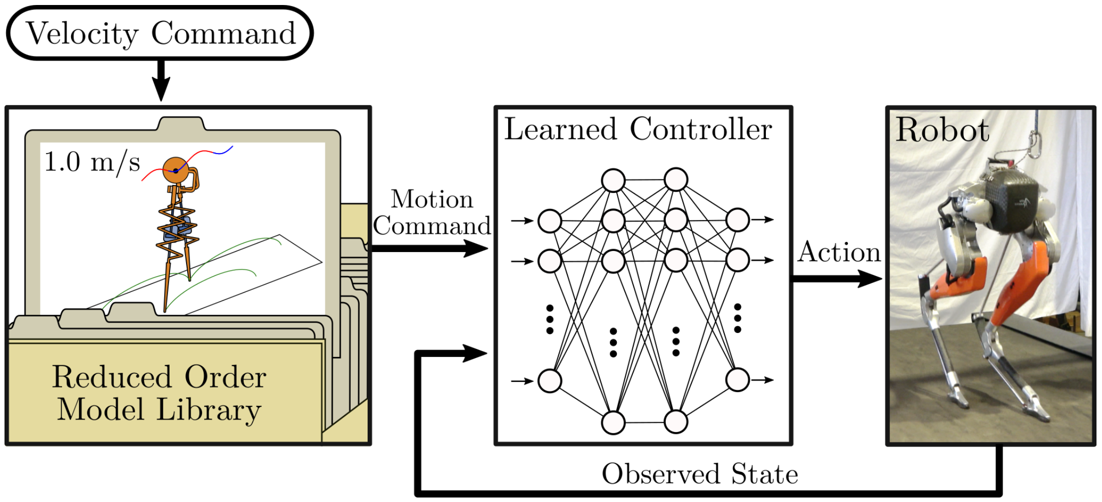


Figure 7.1: Our proposed control hierarchy which demonstrates a learned controller for a legged robot that is commanded using reduced-order model motions. In future work, this library of reduced order model motions can be replaced by a dynamic motion planner.

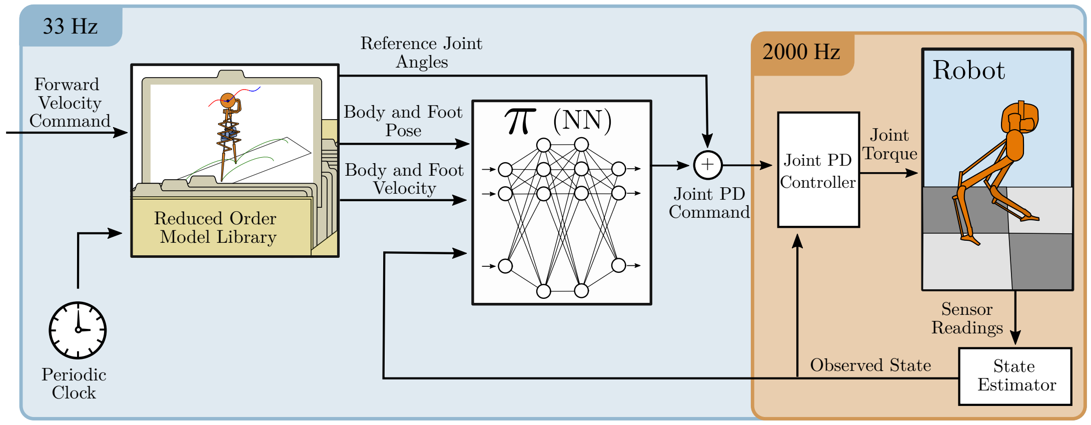


Figure 7.2: The control diagram for both learning in simulation and running on hardware. At a relatively slow 33 Hz, the library is sampled and the learned policy is evaluated. At a much faster 2000 Hz, the joint PD controller is evaluated, the commanded torques are sent to the motors, and the state estimator is updated. The inputs to the learned policy are only the reduced-order model motion and the state estimate.

as a goal [128, 100], which motivates our choice to use an actuated variation of the SLIP model in this letter.

In this paper we present a method of using reduced-order models of walking to direct high quality, transferable walking controllers and demonstrate its effectiveness on a Cassie series robot from Agility Robotics. We use the bipedal actuated SLIP model as the reduced-order model of walking to create a library of gaits across different speeds. The gaits are optimized using a direct collocation method to create energetically optimal walking cycles. Following these trajectories makes up 70% of the reinforcement learning reward while 20% is for foot orientation and 10% is for smooth actions. The resulting controller produces visually natural motions with clear correspondence to the reference trajectories. These results show that reduced-order model trajectories are useful tools in training and controlling learned walking controllers.

7.3 Background

Reinforcement Learning is a learning framework in which agents learn what actions to take in order to maximize their cumulative future reward. Policy gradient methods, such as Proximal Policy Optimization (PPO) [136], are a popular choice of reinforcement learning algorithms that have been successfully applied to generate control policies for robotic systems, including legged robots [161, 149]. An important part of using reinforcement learning to solve complex control problems is the design of the reward function evaluated after each agent-environment interaction. Most applications of reinforcement learning to legged locomotion employ heuristic reward functions to produce walking behavior [80, 149]. Though this method has been effective, it is often hard to describe a desired motion through objective, generic reward functions. The definition must be sufficiently detailed to prevent maladaptive policies from learning motions that exploit features of the reward or simulator and do not accomplish the underlying goal when transferred to hardware. Researchers often use long and complex reward functions to prevent these maladaptive policies; [80] uses 8 different reward terms for a single walking task.

A different approach is to use a single expert trajectory as a reference motion [161]. The reward function encourages motions that are close to the specified expert trajectory, which can result in a policy that closely recreates the desired motion. The use of the reference information discourages exploitative policies since the desired motion is now densely and properly represented. When a reference motion reward makes up a significant portion of the total reward, it severely restricts the space of solutions to be those near the reference motion. For some tasks this restriction is unacceptable; however, for bipedal locomotion restricting the final behavior to resemble a normal walking gait is actually preferable because it disincentives strange, exploitative behaviors.

We note that the method from [161] used a single reference trajectory. To provide effective information for a variety of speeds, this single trajectory was "stretched" and "compressed" to higher and lower speeds, sometimes creating trajectories that were physically infeasible. These infeasible trajectories may harm the learning problem by making the trajectory matching reward signal conflict with the dynamics of the system. Our work mitigates this conflict by using reduced order model trajectories with inverse kinematics to produce feasible walking trajectories for every training speed. Furthermore, the use of a reduced-order model allows for greater flexibility in the control system, by allowing future work to use the reduced-order model trajectories as a form of higher level planning.

7.4 The Control Hierarchy

We created a control hierarchy (Fig. 7.2) to allow us to train and test a walking controller that utilized reduced-order model motion. The only external input to the system is a human operator's forward velocity command. Internally, a periodic clock increments forward through the walking cycle. The velocity command and clock are inputs to a library of reduced-order model motions (§7.4.1). The library returns the positions and velocities of the reduced-order model's body and feet for use as input to the learned policy. Additionally, the library contains a set of robot-specific motor angles that correspond to a robot pose that match the body and foot positions. The learned policy is evaluated and the output is summed together with the reference joint angles to form the motor proportional-derivative (PD) command. These commanded angles are sent to the high frequency control loop (§7.4.2). This control loop evaluates the PD controller, sends torques to the motors, measures robot sensors, filters sensor data, and estimates the full state of the robot. This structure is utilized not just in hardware but also in the simulation environment we use to train the learned policy (π).

7.4.1 The Reduced-Order Model Library

Our motion library consists of task space (body and foot) trajectories for periodic walking over a range of forward speeds. Body and foot trajectories will be used as an input to the learned policy as well as the majority of the reward signal. These trajectories will not be strictly dynamically feasible on the full order robot, but they should describe a nearly feasible center of mass motion that can be produced by valid ground reaction forces at the feet. To create each trajectory we optimize the reduced-order model and augment it with a minimum-jerk swing leg profile. We additionally calculate motor angles for each motion through offline inverse kinematics to use as a feed forward term into the PD controller.

The motions in the library are energetically optimal periodic gaits of the 3D, bipedal actuated SLIP model. This model consists of a point mass body and two massless legs (Fig. 7.3). Parameters of the model were chosen to closely resemble the Cassie robot, see Table 7.1. Each leg has a extensible actuator in series with a spring and a damper. The trajectory optimization method we use is a direct collocation method where the state and inputs are discretized and dynamics are enforced through equality constraints between sequential states [117]. Our implementation is not contact-invariant so we needed to specify the contact sequence. We define a walking contact sequence made up of alternating single stance and double stance phases. We vary the average forward velocities from 0 to 2 m/s in steps of 0.1 m/s as an equality constraint on the final state. The energetic objective

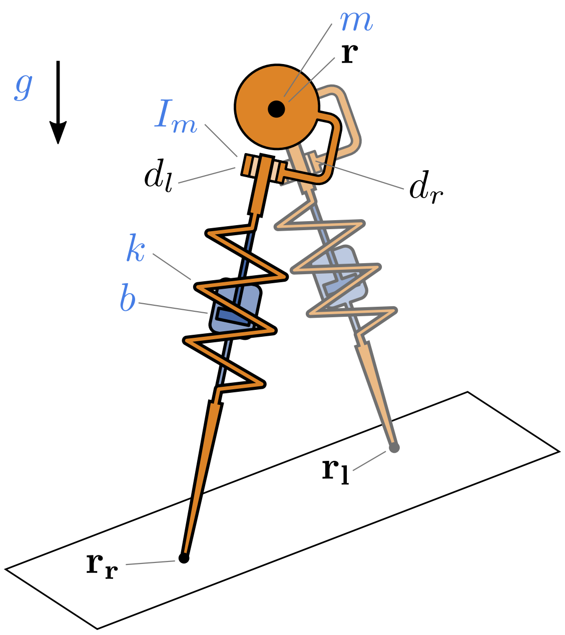


Figure 7.3: The 3D, bipedal actuated spring loaded inverted pendulum model which we optimize to create walking gaits for our motion library. This model has a point mass body with no rotational inertia and two massless, compliant, actuated legs.

	Symbol	Value and Unit	Description
States	$\mathbf{r}$	- $[m,m,m]$	3D body position
	d_l	- $[m]$	Left leg actuator setpoint
	d_r	- $[m]$	Right leg actuator setpoint
Parameters	m	30 $[kg]$	Body mass
	k	3000 $[N/m]$	Leg spring stiffness
	b	2 $[Ns/m]$	Leg spring damping
	I_m	10 $[kg]$	Leg actuator linear inertia
	g	9.81 $[m/s]$	Gravitational acceleration
Continuous	u_l	- $[m/s^2]$	Left actuator acceleration
Inputs	u_r	- $[m/s^2]$	Right actuator acceleration
Discrete	$\mathbf{r_l}$	- $[m,m,m]$	Left foot stance position
Inputs	$\mathbf{r_r}$	- $[m,m,m]$	Right foot stance position

Table 7.1: States, parameters and control inputs for the actuated SLIP model used to generate the library of motions.

function we use is

$$f = \int_0^T (u_l^2 + u_r^2)dt \tag{7.1}$$

where u_l and u_r are the accelerations of the leg actuators. This cost function represents the resistive losses in an electric motor when it applies forces to accelerate the inertia of its rotors. We ensure kinematic feasibility by including a conservative constraint on the maximum and minimum leg extension. To generate the swing foot motions we calculated minimum integrated jerk squared motion profiles. These profiles connect the footholds from the trajectory optimization with a specified vertical clearance height at the midpoint. We use a modified version of COALESCE [84] to generate the problem and its analytical gradients and IPOPT [158] to solve the problem.

An example optimal motion with a mean velocity of 1.0 m/s is shown in Fig. 7.4. The different phases can be seen in the main body's path where it reaches its peak height in single stance and the minimum height in double stance. The feet follow smooth paths with the 0.2 m specified vertical clearance height.

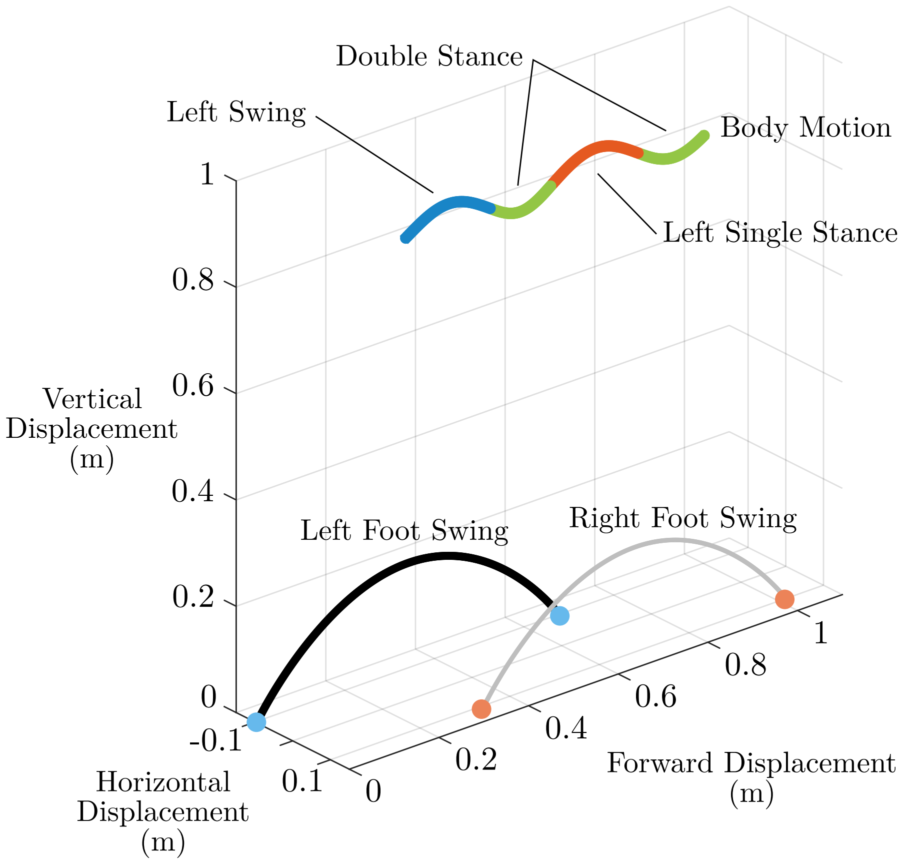


Figure 7.4: The optimized task space motion of the reduced-order model for 1.0 m/s walking. The hybrid phase is shown through the color of the body trajectory. This motion together with its velocities make up the motion library.

7.4.2 High Frequency Control Loop

The task space motions of the reduced-order model are inputted to a learned policy running at 33 Hz which outputs delta motor positions δ_a. These deltas are added to the baseline joint angles from the library, $\hat{a}$, to create the final command for the PD controller, $a = \hat{a} + \delta_a$. The use of joint angle baseline actions was used in prior work on learned walking controllers for Cassie [162]. The actions are passed to a joint level PD controller with fixed feedback gains running at 2 kHz. Note that the learned policy only outputs position targets while the velocity targets are always set to zero. This means the proportional-derivative controller should more accurately be called a proportional-damping controller. We choose this structure as previous work has shown learning PD targets to be easier and produces higher quality motions [114].

7.5 Reinforcement Learning

The optimal learned controller should be able to capture the most important features of the reduced-order model's motion and translate them into a stable walking behavior that functions on hardware.

7.5.1 Problem Formulation

The inputs to the policy are the estimated robot state and the desired positions and velocities of the robot's pelvis and feet. The estimated robot state contains the pelvis height, orientation, translational velocity, rotational velocity, and translational acceleration in addition to the positions and velocities of the actuated and unactuated joints on Cassie. This state estimate together with the commanded pose and velocity from the reduced-order model library form a $64D$ input space. The output of the learned policy is a $10D$ vector containing motor position targets for each of Cassie's actuated joints.

Parameter	Value
Adam learning rate	1×10^{-4}
Adam epsilon	1×10^{-5}
discount (γ)	0.99
clipping parameter (ϵ)	0.2
epochs	3
minibatch size	64
sample size	5096

Table 7.2: PPO Hyperparameters

We learn control policies by using a simulated model of Cassie[1] in the MuJoCo Physics simulator [152]. Our dynamic model of Cassie includes the reflected inertia of each motor (defined as "armature" in MuJoCo). We also attempt to model actuator delay by limiting when new torque commands are actually executed. Desired torques only take effect 0.003 seconds (6 time steps of the high frequency, PD control loop) after being "sent" to the simulator. We believe these extra facets of the model help improve the policy's robustness against differences in simulation and reality, enabling cleaner sim-to-real transfer.

An important part of this setup is that even during training we use an *estimate* of the state rather than the true state. Though we have access to the true simulated state we instead pass the simulated sensor values into a state estimator to get a simulated "observed state." This incorporates simulated sensor noise and state estimator dynamics into the learning process, which is an essential part of making policies robust enough for sim-to-real transfer. This allows us to use the exact same controller structure on hardware, effectively just switching out the simulated robot for the real robot.

7.5.2 Learning Procedure

The reinforcement learning algorithm we use is an implementation of PPO[2] with parallelized experience collection and input normalization[136]. Our policy is a fully connected feed-forward neural network with 2 hidden layers of 256 nodes each. We choose to use fixed covariance instead of making it an additional output of the policy. The hidden layers use the ReLU activation function and the output is unbounded. This architecture was chosen because previous work found it was large and deep enough to generate high quality locomotion across a range of walking speeds [161]. More information on training hyperparameters can be found in table 7.2. At the start of each episode, a reference trajectory is randomly selected from the reduced-order model library and the simulated model of Cassie is set to a random starting position in the trajectory's walk cycle. A single step of agent-environment interaction includes the policy computing an action, sending it to the low-level PD controller which simulates forward 1/33 of a second, and retrieving the next state in the 33 Hz execution cycle. We define the maximum episode length for this MDP problem to be 400 steps of agent-environment interaction, which corresponds to 12 seconds. Episodes are terminated when the number of steps reaches the maximum episode length or when the reward for the current step is less than 0.3. This termination condition encapsulates when the robot falls to the ground or if it deviates excessively from the goal behavior.

We design the following reward function that is evaluated after each step:

$$\begin{aligned} r = \quad & 0.3\, r_{\text{CoM\_vel}} + 0.3\, r_{\text{foot\_pos}} + 0.1\, r_{\text{straight\_diff}} \\ & + 0.2 r_{\text{foot\_orient}} + 0.1 r_{\text{action\_diff}} \end{aligned}$$

All of the terms in the reward function are computed as the negative exponential of a distance metric. This lets us limit the maximum reward per step to 1 and per

[1]Simulation and state estimation library available at `https://github.com/osudrl/cassie-mujoco-sim`

[2]Reinforcement Learning code, reward functions, and ASLIP reference motions available at `https://github.com/osudrl/ASLIP-RL`

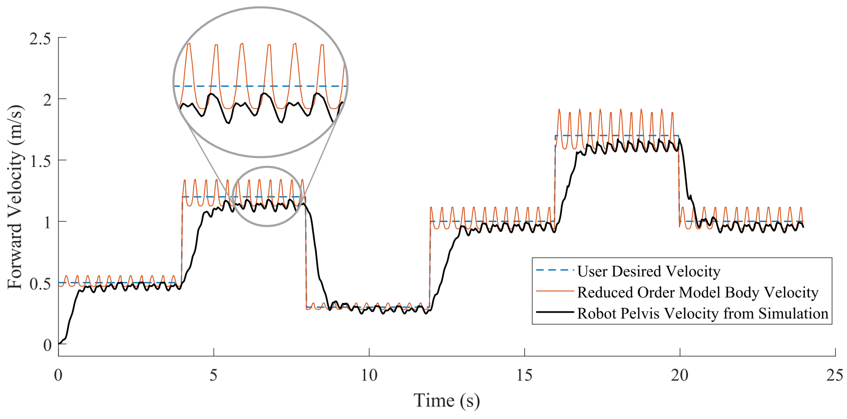


Figure 7.5: Forward velocity comparison of the user desired velocity, the corresponding reduced-order model center of mass velocity, and the simulated robot's pelvis velocity. As can be seen, the policy closely tracks the user's desired velocity. The difference between the reduced-order model and robot's velocity show the learned policy does not emulate the spikes in reduced-order model's body velocity which occur at touchdown.

episode to 400.

The first three terms of the reward function account for 70% of the total reward. Together they penalize the differences between the current state of the robot and reference task space position and velocity. The center of mass velocity matching reward, defined as

$$r_{\text{CoM\_vel}} = \exp(-||\mathbf{v}_{\text{CoM}} - \mathbf{v}_{\text{refCoM}}||), \tag{7.3}$$

incentivizes matching the robot's center of mass (pelvis) velocity to the reference velocity. However, $r_{\text{CoM\_vel}}$ is calculated using the *local* pelvis frame which prevents the policy from receiving a large reward for sidestepping or walking diagonally. To ensure the robot locomotes in the forward direction, $r_{\text{straight\_diff}}$ rewards the lateral robot position being close to zero.

To get the controller to track the reference foot positions, $r_{\text{foot\_pos}}$ rewards the robot's foot positions to be close to the reference motion's foot positions, where the foot position is defined as the position of the foot relative to the body. On the full order robot, the orientation of the foot joints is important for stable walking on hardware. However, our reduced order model has point feet which do not describe or incentivize any particular foot orientation. We reward forward pointing toes and feet parallel to the ground through $r_{\text{foot\_orient}}$.

A reward term that helps the transfer to hardware is $r_{\text{action\_diff}}$, which penalizes the distance between the last action and the current action and results in smoother action outputs. Without this term the policy can converge to behaviors that rapidly oscillate the commanded motor angles which is not conducive to success on hardware.

It is important to note that with the inclusion of the $r_{\text{action\_diff}}$ term, reaching the maximum reward is impossible and not an expectation, as the policy would need to output a constant motor angle while tracking the reference motion. Furthermore, perfectly matching the positions of the reduced-order model at each step is likely not possible because of the significantly more complex dynamics of the full order robot. As seen in [128, 100], directly applying spring-mass behavior to a robot is challenging and sensitive. Thus our learned controller should use the spring mass model as a guide towards highly effective walking solutions that work for the robot on hardware.

7.6 Results

We trained ten different policies from randomized initial weight seeds for our method. The training process takes just under five hours of wall clock time using 50 cores on a dual Intel® Xeon® Platinum 8280 server. The policy learns to step in place after about 25 million timesteps, and converges to a reward of almost 300 after about 175 million timesteps, where it is able to track all of the walk cycles in the reduced-order model library.

7.6.1 Simulation

By varying the user-provided forward velocity to the reduced-order model library, we demonstrate the learned control policy's ability to smoothly transition between

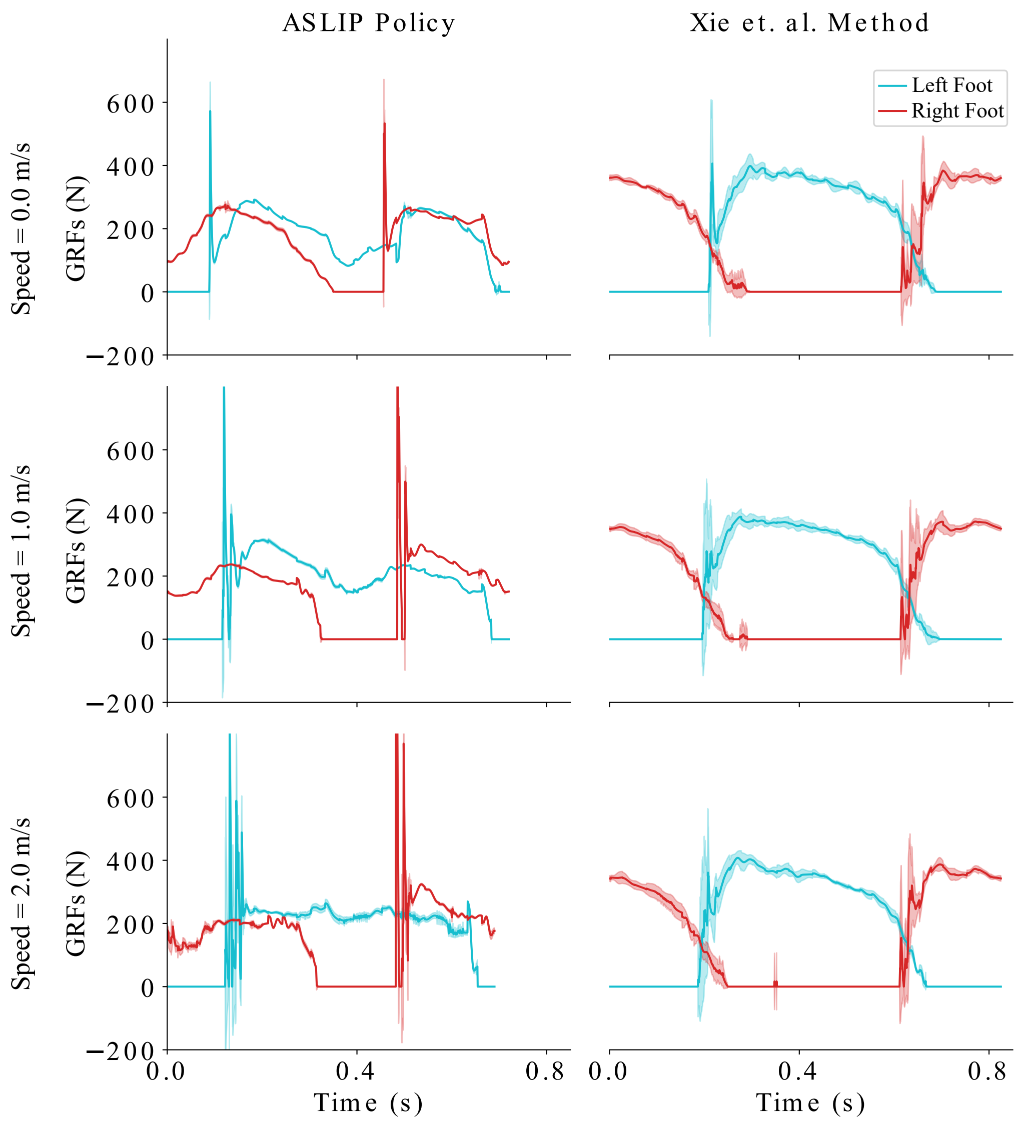


Figure 7.6: Ground reaction force profiles of the actuated SLIP policy compared to a single reference trajectory policy for [161] method. The actuated SLIP policy shows double hump ground reaction forces as is expected from spring mass walking, where the single reference policy shows flattened single hump ground reaction forces similar to linear inverted pendulum walking.

discrete reference walk cycles (Fig. 7.5). The policy produces walking behavior with oscillations in pelvis velocity that correspond with the oscillations in center of mass velocity from the reduced-order model. Furthermore, this velocity tracking succeeds across the broad set of commanded speeds and the transitions between them.

In order to quantify how well desired foot locations can be realized through this control hierarchy, we measure the average error between the foot touchdown locations of the reduced order model commands and the robot in simulation (Fig. 7.7). At all speeds, we observe that the robot places its feet slightly wider than the reduced-order model. Above 1.0 m/s, we see that the robot's footsteps lag more and more which corresponds to error in tracking the reduced-order model's forward velocity. At speeds under 1.0 m/s this placement error doesn't exceed an average of 10 cm.

The ground reaction forces show that our policy produces features indicative of spring mass walking that were not explicitly incentivized by the reward function. We compare the ground reaction forces across different commanded speeds to those from the single reference trajectory policy (Fig. 7.6). Particularly at 0 and 1.0 m/s the actuated SLIP policy has a double hump ground reaction force which is present in spring mass walking. In comparison, the single reference trajectory has a single hump ground reaction force with a flattened peak for all speeds. This type of ground reaction force is expected from a bipedal walking policy that holds the body at a constant height, similar to the linear inverted pendulum [85]. These ground reaction forces are not themselves a useful measure of performance of the gait, but instead provide us evidence that our learned controller is emulating the dynamics of our reduced order model.

7.6.2 Hardware

We directly transfer the policies trained in simulation to hardware, demonstrating that this approach can achieve a strong sim-to-real transfer. We observe that the the learned walking motion is springy, with slight oscillations in the pelvis

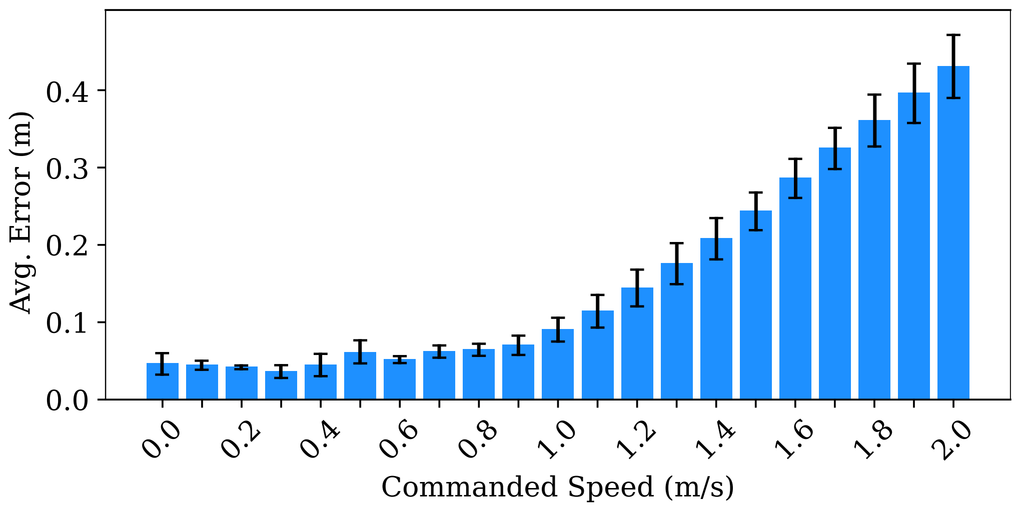


Figure 7.7: Average foot placement error and standard deviation over 15 foot steps at various speeds. The average error across speeds under 1.0 m/s is relatively low and is sufficient for planning precise footstep locations.

velocity and changes in leg length directly corresponding to the same variations in the motion of the reduced-order model. This motion correspondence can be seen in Fig. 7.8 and our accompanying video, which shows Cassie walking using our learned policy for an extended period of time, as well as a comparison between the motions of the reduced-order model, the learned controller in simulation, and the learned controller on hardware. The video also shows that the stepping frequency of all three stages in the control hierarchy: reduced-order model, simulation, and hardware, match for the same forward velocity commands.

To test the ability of these policies to rapidly change speeds in hardware, the human operator sends the robot sudden changes in velocity commands. The results of this trial are shown toward the end in the attached video. This controller is able to walk significantly faster and with a longer stride than was possible using previous model-based control methods on the same robot [11].

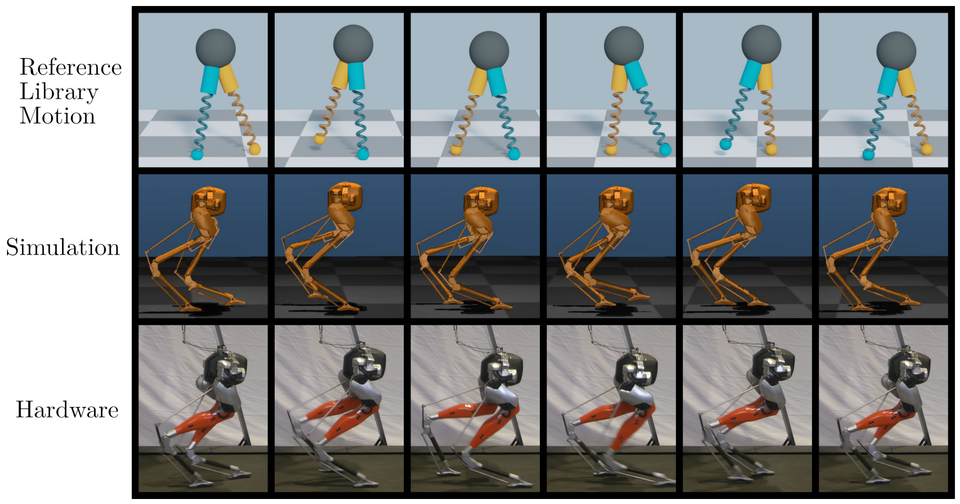


Figure 7.8: Motion comparison of the reduced-order model, simulation, and hardware at different phases in the gait. The top images represent the desired reference motion to recreate, the middle images show the learned policy in simulation, and the bottom images show the learned policy executed on hardware.

7.7 Conclusion

In this letter, we have presented an effective control structure for producing spring mass-like motion on a human scale bipedal robot. This method employs reduced-order model reference trajectories to inform the learning process of the desired task space motion. We find that this method is successful in producing similar motion to the actuated SLIP model and generates policies that can realize this behavior on the bipedal robot Cassie. We found success using the actuated SLIP model as the reduced order model to guide Cassie. One should consider carefully the choice of reduced order model when applying this work to other robots. Continuations of this work will focus on extending the variety of motions the low-level policy can track and improving the foot step location tracking.

This low-level controller will enable many different opportunities for integration of high-level motion planners. Now that we have a policy capable of following a desired reduced-order model motion, we can work to extend this to generate policies that follow any arbitrary reduced-order model trajectory. This would allow for incorporating a high-level planner in the reduced-order model space, such as the planner proposed in [30]. Allowing for reactive planning decisions like navigation and obstacle avoidance to happen at the reduced-order model level will also make it significantly easier to achieve fully autonomous agile legged robots.

Acknowledgments

We thank John Warila and Dylan Albertazzi for their assistance rendering videos, Helei Duan, Jonah Siekmann, Lorzeno Bermillo, and Pedro Morais for productive discussions.

Chapter 8: Learning Transient Locomotion through Centroidal Momentum References

This chapter contains work aimed at performing agile, transient maneuvers. The first component is to build a system to reliably optimize transferable trajectories for a centroidal momentum model. Second, we learned locomotion controllers which are tasked with performing these agile maneuvers. We investigated the utility of centroidal momentum references compared to center of mass only references (pure inverted pendulum models) and reference free learning as well as a novel epilogue reward. This chapter contains work that formed two IEEE Humanoids papers but is tightly related so it is combined into a single chapter. Much of the text is reproduced from the conference paper submissions with expanded sections and reorganization as was necessary.

Contributions

Kevin Green devised the approach, contributed to the optimization implementation, co-wrote the manuscript and supervised the investigation. Ryan Batke led the optimization implementation and learning implementation and co-wrote the manuscript. Fangzhou Yu led the maneuver reinforcement learning implementation and co-wrote the manuscript. Jeremy Dao contributed to the learning implementation and co-wrote the manuscript. Ross L. Hatton, Alan Fern, and Jonathan Hurst supervised the investigation and edited the manuscript.

Optimizing Bipedal Maneuvers of Single Rigid-Body Models for Reinforcement Learning

Ryan Batke, Fangzhou Yu, Jeremy Dao,
Jonathan Hurst, Ross L. Hatton, Alan Fern, and Kevin Green

(In Review) IEEE Humanoids 2022

Dynamic Bipedal Maneuvers through Sim-to-Real Reinforcement Learning

Fangzhou Yu, Ryan Batke, Jeremy Dao,
Jonathan Hurst, Kevin Green, and Alan Fern

(In Review) IEEE Humanoids 2022

8.1 Abstract

For legged robots to match the athletic capabilities of humans and animals, they must not only produce robust periodic walking and running, but also seamlessly switch between nominal locomotion gaits and more specialized transient maneuvers. Despite recent advancements in controls of bipedal robots, there has been little focus on producing highly dynamic behaviors. First, we propose a method to generate reduced-order model reference trajectories for general classes of highly dynamic maneuvers for bipedal robots for use in sim-to-real reinforcement learning. Our approach is to utilize a single rigid-body model (SRBM) to optimize libraries of trajectories offline to be used as expert references in the reward function of a learned policy. This method translates the model's dynamically rich rotational and translational behavior to a full-order robot model and successfully transfers to real hardware. Within this work we introduce a set of transferability constraints that amend the SRBM dynamics to actual bipedal robot hardware, our framework for creating optimal trajectories for a variety of highly dynamic maneuvers as well as our approach to integrating reference trajectories for a high-speed running reinforcement learning policy. Second, we investigate the reinforcement learning problem on nominal locomotion using the SRBM reference and on dynamic four-step turns. On the bipedal robot Cassie on which we were successfully able to demonstrate highly dynamic grounded running gaits up to 3.0 m/s. Inspired by conventional optimization-based control techniques for legged robots, this work applies a recurrent policy to execute four-step, 90 turns trained using reference data generated from optimized single rigid body model trajectories. We present a novel training framework using epilogue terminal rewards for learning specific behaviors from pre-computed trajectory data and demonstrate a successful transfer to hardware on the bipedal robot Cassie.

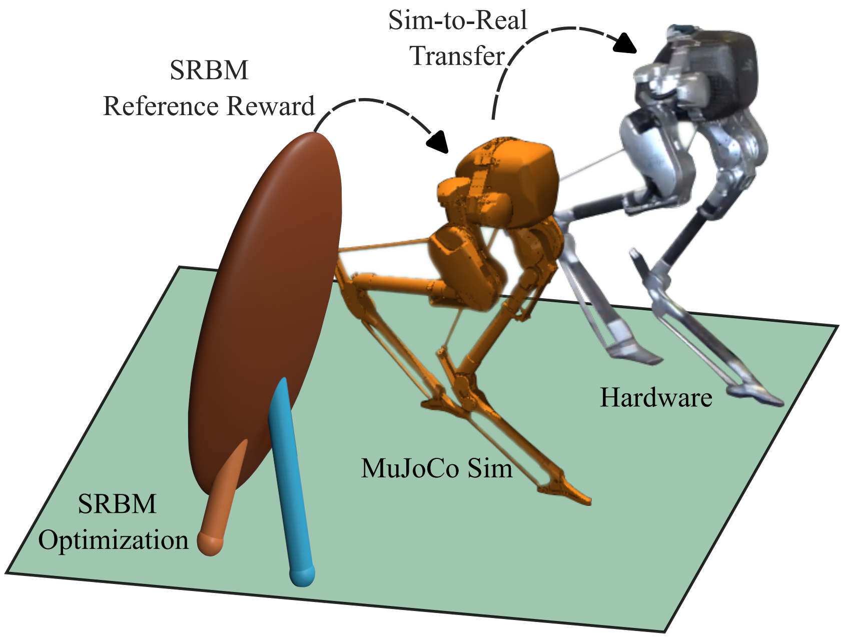


Figure 8.1: Dynamic maneuver workflow - SRBM optimized reference trajectories are used to develop learned controllers in simulation that are deployed on hardware.

8.2 Introduction

Bipedal animals from ratites to humans are capable of executing dynamic and aggressive motions that can seem effortless and graceful, but in reality require a complex balance of body momentum with fast and precise footstep placements. Human athletes and animals are able to execute maneuvers where in a few steps they redirect their momentum in sharp, often successive turns to quickly change direction. Additionally, both humans and many animals are capable of leaping into the air to reach higher locations, cross gaps or as a means to change direction.

Bipedal robots have recently demonstrated increasingly dynamic capabilities [142, 80]. However, these advancements are still far behind the feats achieved by well tuned biological systems. Modern control techniques that produce reliable locomotion such as model predictive controllers (MPC) based on Linear Inverted Pendulum (LIP) dynamics [137] severely restrict a systems behavior, preventing the realization of many high speed gaits, highly efficient gaits or dynamic maneuvers.

The usage of reduced-order-models (ROM) is a hallmark of optimization-based bipedal locomotion controllers. These models serve to embed controllers with simplified descriptions of locomotion dynamics to reduce the complexity of planning. Open-loop planning techniques like trajectory optimization (TO) can be used with a ROM to quickly generate highly non-linear optimal trajectories by enforcing meaningful constraints and objective functions on the underlying model. The popular LIP model as well as the Spring Loaded Inverted Pendulum (SLIP model) lack the fidelity to capture the effects of angular momentum on the system that are necessary to achieve maneuvers such as sharp turns and spin jumps [56, 85]. Centroidal Momentum Models are another popular class of ROM commonly used for full humanoid robots [38, 159]. These models characterize both the linear and angular momentum of a robots CoM, but do not encode a mean orientation for the robot which is necessary for our target maneuvers.

Data-driven methods offer a competing paradigm for control. Recent successes with model-free deep reinforcement learning (DRL) based controllers for legged robots have demonstrated a wide range of robust and dynamic behaviours on hardware [142, 96, 40]. These learned controllers have the advantage of being trained in simulation on full-order robot models that contain dynamic information missing from ROMs. Techniques such as dynamics randomization allow for a DRL agent to experience and adapt to various conditions over millions of iterations. This allows for the training of incredibly robust policies that can reliably traverse the sim-to-real gap. As powerful as these techniques have proven to be at demonstrating common gaits, the extension of reference-free DRL to structured maneuvers is still an unsolved problem.

A promising alternative to relying solely on either ROM based optimizations or reference-free DRL is to utilize a dynamically rich model offline to first plan highly non-linear behaviors through TO. These reference trajectories can then be used as a central part of a reward function to allow a learned controller to generalize the optimal trajectories from a simple model to the full-order robot and transfer to hardware.

In this work we present a single rigid-body model (SRBM) as applied to the bipedal robot Cassie. Using TO we create libraries of optimal trajectories offline that utilize the SRBM to plan a variety of highly dynamic behaviours such as running routes and jumps. These trajectories are then incorporated as part of a policies reward function and trained using proximal policy optimization (PPO) to develop controllers that are capable of executing the desired maneuver online on real hardware. The challenge of successfully transitioning between different policies is addressed during the training process with the concept of an epilogue terminal reward. To test the viability of our proposed technique implemented on real world hardware, we demonstrate the successful sim-to-real transfer of Cassie performing a four-step 90 degree right turn using a policy trained with trajectory data that successfully transitions between another policy performing a running gait developed from our previous work.

8.3 Background

8.3.1 Reduced-Order-Models

A SRBM approximates the inertia of a legged robot's many rigid bodies to a single body located at the center of mass (CoM) which is manipulated by ground reaction forces (GRFs) applied at foot locations through an idealized, massless leg. The power of the SRBM is in its simplicity, using only a small number of parameters it is able to characterize the linear and rotational dynamics that are required to achieve highly aggressive maneuvers. These models have been applied to quadruped robots to great success, allowing for hardware demonstrations of dynamic behaviours such as flips [86, 1]. SRBMs have been utilized for dynamic quadrupeds in conjunction with both optimization [18] and RL based controllers [163].

Similar application of SRBMs are much less frequent in the domain of bipeds. This is because, unlike quadrupeds, bipedal robot's legs often represent a more significant contribution to the robot's inertia [144]. A SRBM can still be used to plan for less-ideal bipeds and full humanoids, but requires careful consideration of how to best to apply the ROM to the full-order robot. An impressive example of the SRBM being utilized to plan for angular momentum rich trajectories on a biped is Boston Dynamics' Atlas. Atlas can perform a wide variety of maneuvers from backflips to sequential parkour jumps onto slanted surfaces [22]. Boston Dynamics has indicated that TO is used offline to create libraries of task-specific maneuvers based off of an SRBM that is adjusted for kinematic feasibility [94, 42]. Online the robot uses MPC guided by the offline optimal trajectories to perform short horizon optimizations that make real-time adjustments allowing for the execution of the dynamic behaviours on hardware.

Previous work combined reference trajectories and DRL to develop walking controllers for Cassie [161], but these were based off only a single manipulated reference and thus were limited in application. Additional work was conducted using SLIP models to create libraries of reference trajectories to guide DRL [58]. Most similar to this current work, combining TO and DRL for locomotion was shown by [23] for a terrain-aware quadrupedal robot.

8.3.2 Learning for General Locomotion

RL has shown to be a promising alternative to model-based control of legged robots [141, 149, 96]. However, most published work on the application of RL to legged locomotion focuses on performing cyclic gaits, while in this work we are concerned with more dynamics one-off maneuvers. Prior work on performing different behaviors with learned methods use RL to train a singular policy to execute all desired behaviors instead of training separate policies for each individual behavior. This causes the behavior space of the policies to be limited by the richness of a singular reward function, making them ill-suited to learning multiple different behaviors. This work addresses this issue by switching between policies trained to execute specific behaviors. [141] demonstrated a learning framework capable of reproducing all common bipedal gaits for Cassie on a single policy that does not use expert reference trajectories. The resulting policy could continuously transition between different bipedal gaits by adjusting a left and right foot cycle offset parameter. More recent work [48] used a similar learning framework to the one in this work to develop policies for Cassie to step on target footholds. The foothold targeting policies were allowed to modulate the gait behavior to achieve strides of varying length by adaptively changing the stepping frequency. Model differences between simulation and the real world robot were resolved by training the policy with dynamics randomization [149], which has shown to improve the consistency of successful sim-to-real transfers of recurrent neural network (RNN) walking policies for Cassie [143].

8.3.3 Learning for Multiple Behaviors

RL has also been used to learn different locomotion skills as well as smooth transitions between them. [63] demonstrated successful sim-to-real transfer of switching between forward and backward walking on a single policy trained with atomic, task-specific reward functions. Similarly, learned locomotion control policies have shown to be capable of assuming different gait behaviors to negotiate terrain obstacles and gaps. [68] demonstrated the emergence of robust obstacle clearing behavior for torque-controlled legged agents by training control policies using simple reward functions in obstacle-rich simulation environments. These examples of prior work engineer the agent-environment interactions to encourage the emergence of multiple behaviors, which suffer from the same drawbacks as the examples in Section 8.3.2 because multiple behaviors are learned on a single policy.

8.4 Single Rigid Body Model Formulation

Our representation, dynamics and quaternion integration method are inspired by [81]. This body has a configuration space of $SE(3)$ and a velocity in $\mathbb{R}^6$. While this is the precise structure of the space, we elect to use a common representation of the configuration space as $[\mathbf{p_c}, \mathbf{q}] \in \mathbb{R}^3 \times \mathbb{S}^3$. Here $\mathbf{p_c}$ is the 3D position of the body's center of mass (CoM) and $\mathbf{q}$ is the body's attitude as a quaternion. The body's velocity is represented by the CoM linear velocity $\mathbf{v_c} \in \mathbb{R}^3$ and the body frame angular velocity $\omega \in \mathbb{R}^3$. We then combine the configuration and velocity into a single state vector $\mathbf{x}$ which has dynamics

$$\mathbf{x} = \begin{bmatrix} \mathbf{p_c} \\ \mathbf{q} \\ \mathbf{v_c} \\ \omega \end{bmatrix}, \quad \dot{\mathbf{x}} = \begin{bmatrix} \mathbf{v_c} \\ \frac{1}{2}\mathbf{q} \otimes \hat{\omega} \\ \frac{1}{m}\mathbf{F}_W(\mathbf{x},\mathbf{u}) \\ J^{-1}(\tau_B(\mathbf{x},\mathbf{u}) - \omega \times J\omega) \end{bmatrix}. \tag{8.1}$$

Here $\mathbf{x}$ and $\mathbf{u}$ are the state and control vectors, m is the mass, $\mathbf{J} \in \mathbb{R}^{3\times3}$ is the body frame rotational inertia matrix, $\mathbf{F}_W(\mathbf{x},\mathbf{u}) \in \mathbb{R}^3$ are the external forces in the world frame, and $\tau_B(\mathbf{x},\mathbf{u})$ are the external moments as expressed in the body frame. In the rotational dynamics, $\otimes$ represents quaternion multiplication and $\hat{\omega}$ is the angular velocity as a quaternion with zero scalar part. More details on the rotational dynamics can be found in [81].

Our model is actuated by ground reaction forces which are applied by a massless, ideal leg. These ground reaction forces can be thought of as a perfect force vector applied to the body through the foot's contact point. This can then be transformed into a wrench at the body to integrate the dynamics. In our implementation the model can have one, two or no active foot contacts depending on its specified hybrid mode.

8.5 Trajectory Optimization Formulation

We used direct collocation trajectory optimization implemented in COALESCE [84] and solved using IPOPT [158] to create libraries of dynamically feasible open-loop trajectories offline. Dynamically rich trajectories can be generated though specifying a physically meaningful objective function, a sequence of hybrid modes and constraints that shape the behavior of the model to the dynamic maneuvers that are the goal of this work. By specifying sequences of hybrid phases and applying sets of constraints, through the optimization we are able to develop a wide variety of agile maneuvers that can be easily iterated and shaped to create expert references for learning-based controllers.

8.5.1 Hybrid Modes

We are interested in agile maneuvers that span multiple footsteps which means that our motions will span multiple hybrid modes. While there is much interest in contact implicit TO, we chose to prescribe the contact sequence a priori. We chose this because the space of contact sequences for point foot bipedal models with no arm contact in a flat environment is relatively small. For standard locomotion gaits at a commanded speed there are three reasonable options: The robot can walk with a double stance phase, it can run with an aerial phase, or it can perform a grounded run with no double stance or flight phase. Each of these corresponds to a different sequence of three hybrid modes: aerial flight, single stance, double stance. Our model's massless, ideal legs mean that we can consider left and right single stances to have identical dynamics. In this work we are interested in agile maneuvers, as well as dynamic jumps. For running and turning maneuvers we use a grounded run contact pattern, which consists of alternating single stance modes. For jumping maneuvers we use double stance and aerial flight modes.

8.5.2 Decision Variables

Our TO consists of several hybrid modes serially appended to each other, with a total of M modes for any maneuver. Each of these modes has N collocation points which evenly divide the duration of the hybrid mode in question. This means that for any mode we have N sequential variants of the state and inputs. Our state is parameterized by 13 components: 3 from CoM position ($\mathbf{p}_c$), 3 from CoM velocity ($\mathbf{v}_c$), 4 from the orientation quaternion ($\mathbf{q}$), and 3 from body frame angular velocity (ω). If we are in single-stance we need to collocate the inputs, which in this case are N of the ground reaction forces $F \in \mathbb{R}^3$. In order to adjust the timing of the hybrid

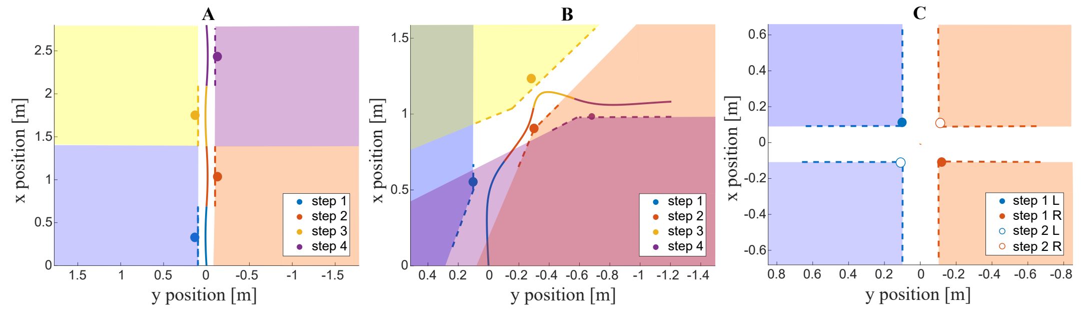


Figure 8.2: Footstep constraint visualizations, A: forward running, B: 90^0 turn, C: 90^0 spin jump. These constraints are built of a pair of linear constraints on foot location in X-Y space. The constraints are based on a “nominal” heading and are relative to the initial and final positions of the body in that dynamic phase.

transitions the optimizer needs to be able to adjust the duration of the phases. To this end, we also create a single decision variable per phase which represents the phase’s duration. Lastly, the optimization needs to be able to adjust the location where the stance foot is placed. This is accomplished by adding a single position vector as a decision variable for single-stance phases $\mathbf{p}_f \in \mathbb{R}^3$. For double stance phases we have two foot positions, $\mathbf{p}_{f,1},\ \mathbf{p}_{f,2}$, and similarly $F_1,\ F_2$. In this work we constrain the vertical components of foot position to be zero for flat ground gaits. This results in a total decision variable count of $13N + 1$ for flight phases, $16N + 4$ for single-stance phases, and $19N + 7$ for double stance.

The indexing $x(m, n)$ is used in this paper to refer to the m^{th} hybrid mode and the n^{th} collocation point of any variable. When no indexing is used this refers to a variable throughout the entire trajectory, for each mode and collocation point. The letters i and F refer to the initial and final modes of any maneuver, where 1, and N are used to indicate the first and final collocation point of any mode.

8.5.3 Transferability Constraints

In order to constrain the model to produce maneuvers which are more likely to transfer to the physical robot we apply some general constraints. First is kinematic feasibility: The workspace of most real robot legs are complicated spaces, particularly so for Cassie's four bar leg mechanism. We would also need to describe this workspace relative to the center of mass of the robot. Knowing the difficulty in creating a very accurate model, we compromise and impose a single constraint on the total leg length,

$$\|\mathbf{p}_c - \mathbf{p}_f\| \leq L_{\max}. \tag{8.2}$$

A constraint is also imposed on the angle that the leg can make with respect to the unit vector pointing directly downwards in the body frame, $\hat{\mathbf{z}}_{\mathbf{body}}$. This is used as a proxy for joint and configuration limits. A variable ψ is used to define the bounds of how different the normalized leg vector can be from $\hat{\mathbf{z}}_{\mathbf{body}}$,

$$\psi < \hat{\mathbf{z}}_{\text{body}} \cdot \frac{(\mathbf{p}_c - \mathbf{p}_f)}{\|(\mathbf{p}_c - \mathbf{p}_f)\|}. \tag{8.3}$$

Next we impose several constraints on the ground reaction forces. We apply a quadratic friction cone constraint,

$$\mu F_z^2 \geq F_x^2 + F_y^2. \tag{8.4}$$

Additionally, we apply a maximum force constraint based on the physical limitations of our actuation

$$F_z^2 + F_x^2 + F_y^2 \leq F_{\max}^2. \tag{8.5}$$

Our simplified model has no actuator dynamics which means it is able to instantly change the foot force which is not realistic. We impose a maximum yank constraint (rate of change on vertical force) $\dot{F}_{max}$ based on analysis of our previously developed controllers for the Cassie robot.

$$-\dot{F}_{\max} \leq \dot{F}_z \leq \dot{F}_{\max} \tag{8.6}$$

In reality we impose the constraint on the finite difference between subsequent collocated inputs, because the force has no intrinsic dynamics. Feasibility constraints are placed on each component of ω to prevent the optimizer from exploiting the simplified models dynamics, this also yields smoother more feasible motions. We apply a simple bound on the angular velocity, $|\omega| \leq \omega_{\max}$.

Lastly, a simple footstep heuristic is used to present the optimization with areas of feasible space to place the models feet. This constraint ensures that feet are placed on the correct side of the body to prevent leg crossing. The heuristic defines acceptable regions by first constructing two half-planes for each step, drawn from the CoM position at foot touchdown and foot liftoff and extended out infinitely along the current heading. These planes are then projected out in the correct side of the body by some minimum distance $\delta_{\min}$. The region encapsulated by the intersection of these planes forms a feasible region where feet may be placed. $\delta_{\min}$ is needed to prevent the optimizer from exploiting monopod dynamics and placing footstep locations directly under the model which does not translate well to actual biped hardware. Fig. 8.2 depicts a visualization of these constraints for several example maneuvers.

8.5.4 Composing Maneuvers

We are able to design agile maneuvers within our TO framework by specifying sequences of hybrid modes and accompanying sets of constraints that shape the optimization to produce the desired behavior. The transferability constraints from Section 8.5.3 are applied in general to each hybrid mode. Additionally, loosely bounded constraints are also placed on the minimum and maximum duration for each phase. These durations control the stepping frequencies for single-stance phases, and the liftoff, flight, and touchdown durations for jumping motions. Any number of maneuvers can be developed using these principles. For breadth we present three formulations for hybrid mode sequences and constraint sets to create trajectories for nominal forward running gaits, dynamic turns, and spin jumps.

8.5.4.1 Forward Running

To develop a grounded running gait we specify sequences of single-stance dynamics and apply the appropriate set of constraints. Constraints are enforced to ensure that the heading remains mostly constant throughout the gait. To enforce constraints on quaternion orientations, we employ a distance function which bounds the angle difference θ_{tol} that two quaternions can have from each other,

$$d(\mathbf{q}_1, \mathbf{q}_2) \leq \theta_{\text{tol}}. \tag{8.7}$$

We use this function to formulate a constraint between the initial orientation $\mathbf{q}(i, 1)$ and a specified desired orientation for forward running $\mathbf{q}_{\text{run}}$, as well as the final orientation $\mathbf{q}(F, N)$ and the same desired orientation. Average velocity constraints are applied in the form of,

$$\frac{p_x(F, N) - p_x(i, 1)}{T} = v_{\text{des}}, \tag{8.8}$$

where T is the total duration of the maneuver. Additionally, zero average velocity constraints are applied in the lateral direction to p_y. Cyclic equality constraints are applied within each phase and enforced on the bodies height, $p_z(m, 1) = p_z(m, N)$ and vertical velocity, $v_z(m, 1) = v_z(m, N)$. Similar constraints are applied to the angular velocity in the y direction, while the x and z angular velocities are mirrored with constraints such as $\omega_x(m, N) = -\omega_x(m, 1)$

For each hybrid mode the body's final orientation $\mathbf{q}(m, N)$ is mirrored about it's sagittal plane. Using the distance function d, this mirror, $\mathbf{q}_{\text{mirror}}(m, N)$ must be within a tolerance θ_{mirror} of the initial orientation of the mode $\mathbf{q}(m, 1)$,

$$d(\mathbf{q}_{\text{mirror}}(m, N), \mathbf{q}(m, 1)) \leq \theta_{\text{mirror}}. \tag{8.9}$$

8.5.4.2 Turning

Turning maneuvers are sequences of single-stance phases with the explicit desire to change heading throughout the maneuver. The desired heading for each mode is updated by incrementing the current heading by the desired total change in heading over the number of steps for the turn. Using our quaternion distance function d we ensure that that the bodies orientation at the end of the maneuver $\mathbf{q}(F, N)$ aligns with the desired orientation for the turn $\mathbf{q}_{\text{turn}}$ within a tolerance θ_{turn}. Desired velocity equality constraints are only enforced on the initial and final collocation points, $\mathbf{v}(i, 1)$ and $\mathbf{v}(F, N)$, as the optimization needs to be free to modulate the bodies velocity throughout the maneuver. Cyclic equality constraints are only placed on the initial and final heights for the maneuver

$$p_z(i, 1) = p_z(F, N). \tag{8.10}$$

8.5.4.3 Spin Jumps

Our example formulation for jumping maneuvers consist of 4 hybrid modes, a double stance for liftoff, followed by two consecutive flight phases, and ending with a double stance for touchdown. Two flight phases are required so that the jump height can be precisely specified at apex.

Multiple constraints are enforced on the models orientation at different stages of the jump. Our distance function d is used on the final orientation at liftoff, $\mathbf{q}(i, N)$, to constrain it to be within some tolerance θ_{liftoff} of a desired neutral starting position $\mathbf{q}_{\text{liftoff}}$. A loose tolerance $\theta_{\text{touchdown}_i}$ is applied using the same quaternion distance constraint at the beginning of touchdown $\mathbf{q}(F, 1)$ to ensure that the majority of the turn is accomplished in the air to achieve the desired rotation $\mathbf{q}_{\text{touchdown}}$. A second tighter tolerance $\theta_{\text{touchdown}_F}$ applied to the final orientation $\mathbf{q}(F, N)$ ensures that the model makes any required corrections to finish close to $\mathbf{q}_{\text{touchdown}}$.

Equality constraints are placed on $v_c(i, 1), v_c(F, N), \omega(i, 1)$, and $\omega(F, N)$ to enforce that the jumps start and end at rest. Lastly, equality constraints are placed at the final collocation point of the first aerial phase (hybrid mode 2). These ensure that $p_z(2, N)$ reaches the desired height, $p_{\mathrm{z}_{\mathrm{des}}}$, and is at the same time at the apex of the jump by forcing $v_z(2, N)$ to 0. Additional constraints can also be employed to ensure displacement of the body in any direction, as would be required to jump up onto a surface or over a gap.

8.5.5 Objective

The objective for each trajectory is the minimization of the squared ground reaction forces and the resulting moments applied at the body. This objective incentivises smoother control inputs which are more transferable to hardware. The objective function is the integral

$$f = \int_0^T \big(u^2(\tau) + M^2(\tau)\big) d\tau, \tag{8.11}$$

which is approximated using trapezoidal integration.

8.5.6 Library Generation

The end goal of the optimization is to construct entire libraries of optimal trajectories for a given maneuver. For the case of grounded maneuvers such as the running gaits and dynamic turns, these libraries are similar optimizations sweeping through a range of desired velocities. For the spin jumps the optimization could be solved for over a range of desired heights or end displacements of the CoM. The purpose of generating entire libraries for maneuvers is to make them more generalizable and help develop more robust learned control policies that can execute the target maneuver on hardware and for a wide variety of conditions.

Much of the success of solving a TO problem lies with good choice of initial conditions. Initial guesses are expertly chosen for the first optimization of a library. Then the solution is used as the initial guess for the next iteration to speed-up con-

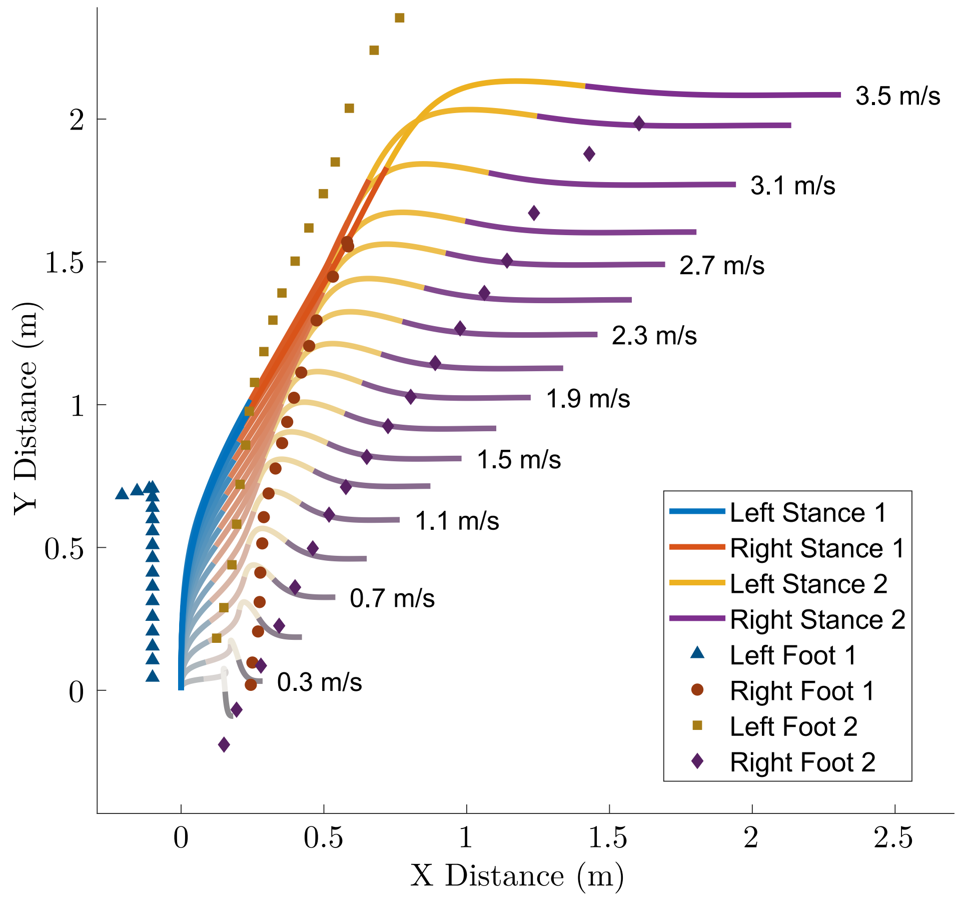


Figure 8.3: Center of mass traces and footstep locations for a library of 4-step 90° turning trajectories.

vergence. This additionally increases the likelihood that the library of trajectories will smoothly vary which is useful for the DRL problem. Fig. 8.3 is an example of trajectories for a library of 4-step 90° turns from 0.1-3.5 m/s.

8.6 Running Reinforcement Learning Problem

Our approach to developing reference guided RL control policies borrows ideas from both referenced-based work [161] as well as more recent reference-free work [141].

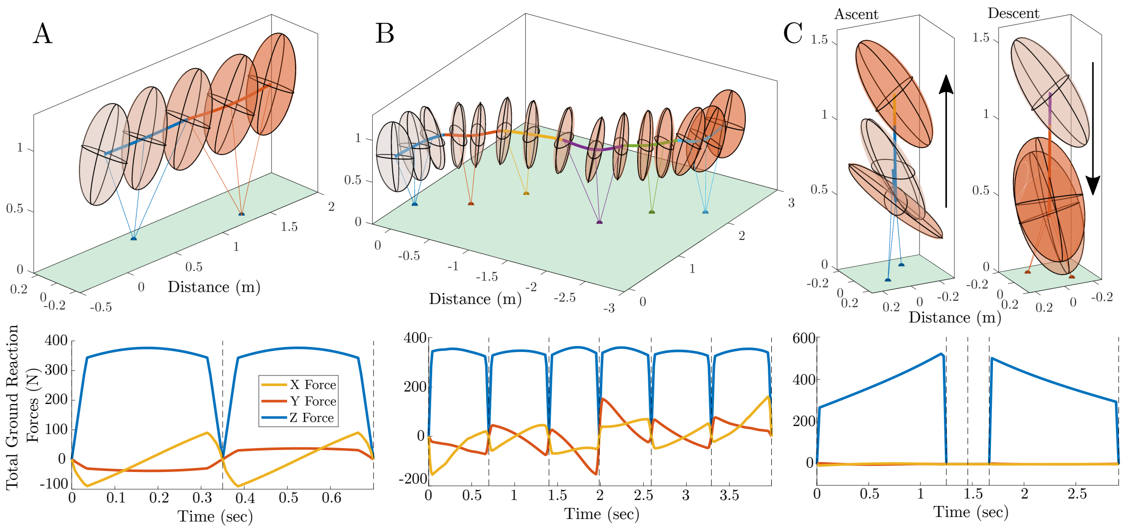


Figure 8.4: Visualizations and accompanying GRF plots for a selection of maneuvers. A: 2-step running trajectory at 2 m/s. B: 3-step $-90°$ turn into 3-step $+90°$ turn at 2 m/s. C: $-90°$ spin jump, CoM reaching 1.2 m. Vertical dashed lines on GRF plots indicate the transition between hybrid phases.

We seek to leverage the rich dynamic information from the SRBM captured in maneuver libraries to create meaningful reward functions that not only bootstrap the learning process from this expert information but are also capable of extending the capabilities of learned controllers to highly dynamic maneuvers. In this work, as a case study to validate the transferability of SRBM dynamics to Cassie we attempt the task of developing a high-speed running policy from a library of optimized running references from 0.1-3.5 m/s. Although running can be described entirely as a periodic phenomenon without ROM information [141], we first seek to learn how well SRBM reference policies cross the sim-to-real gap, and if utilizing this reference information provides any benefits to training over reference-free methods.

8.6.1 Problem formulation

Our control policy is a long-short-term-memory (LSTM) NN with two fully connected hidden layers of 128 units each. The policy takes as input both estimated state and task information. 41 total inputs represent the robot's internal state, comprised of; pelvis orientation, pelvis rotational velocity, estimated foot positions, motor positions and velocities and actuated joint positions and velocities. Three additional inputs provide task information in the form of a commanded velocity and a 2D clock signal. The outputs of the network are the 10 commanded joint PD set points which are sent to high rate, low-level motor controllers.

Control policies are trained using a MuJoCo simulated model of Cassie[1]. The learned controller runs at 40 Hz, with a total of 250 steps per episode (equivalent of 6.25 sec). The reference trajectories range in length from 0.54-1.0 sec and are looped repeatedly for the duration of each episode.

At the beginning of each episode Cassie is initialized with a random state from a previously trained nominal walking gait. This randomization helps to increase the robustness of the policy as it most often needs to recover from poor starting conditions at the beginning of each episode. Additionally the policy is given a random commanded velocity which selects the corresponding reference trajectory.

8.6.2 Learning Procedure

We use proximal policy optimization (PPO) [136] with the Adam optimizer due to its relative simplicity and previously demonstrated successes. Training hyper-parameters provided in Table 8.1.

We formulate our reward function to include components that reward tracking the important dynamics from the reference trajectories. We include terms for matching body orientation (8.12), CoM linear velocities (8.13) and angular momentum $\mathbf{L} = J\omega$ (8.14), as well as x and y components of the foot positions

[1]Simulation available at https://github.com/osudrl/cassie-mujoco-sim

Parameter	Value
Adam discount (γ)	0.95
Adam epsilon	1×10^{-6}
actor learning rate	3×10^{-4}
critic learning rate	3×10^{-4}
gradient update clipping	0.05
batch size	64
epochs	5
sample size	50000

Table 8.1: PPO hyperparameters for single rigid body model reference-based training.

relative to the body (8.15).

$$r_{\mathbf{q}} = 0.05\ \exp(-d(\mathbf{q}, \mathbf{q}^{\text{ref}})) \tag{8.12}$$

$$\begin{aligned} r_v = {} & 0.35\ \exp(-|v_x - v_x^{\text{ref}}|) \\ & + 0.1\ \exp(-|v_y - v_y^{\text{ref}}|) \\ & + 0.1\ \exp(-|v_z - v_z^{\text{ref}}|) \end{aligned} \tag{8.13}$$

$$r_{\mathbf{L}} = 0.15\ \exp(-\|\mathbf{L} - \mathbf{L}^{\text{ref}}\|) \tag{8.14}$$

$$\begin{aligned} r_{p_f} = {} & 0.15\ \exp(-|20 \cdot (p_{f_x} - p_{f_x}^{\text{ref}})|) \\ & +0.15\ \exp(-|20 \cdot (p_{f_y} - p_{f_y}^{\text{ref}})|). \end{aligned} \tag{8.15}$$

Additionally our controller features a piecewise-linear clock signal explained in detail in [40] that rewards the agent based on matching the stepping frequencies determined by the optimization.

$$r_{\text{clock}} = 0.3\ \exp(-|F_{\text{clock}}|). \tag{8.16}$$

Non-reference terms are also added to account for limitations inherent to model

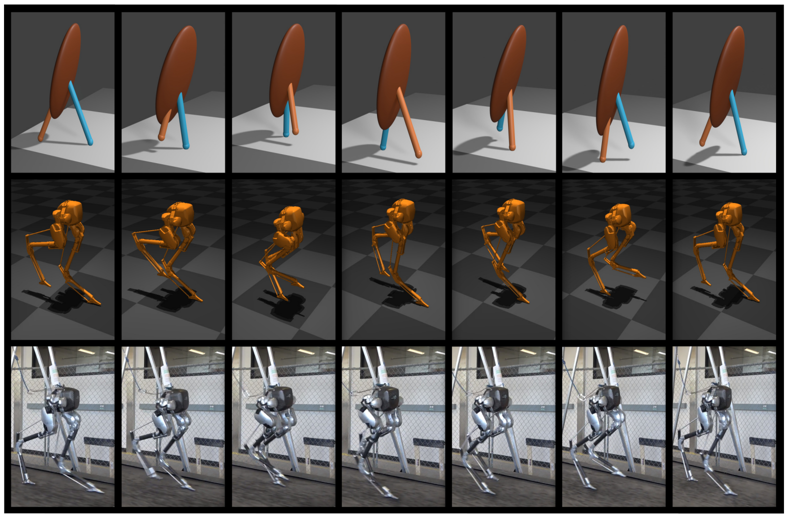


Figure 8.5: Comparison of 3.0 m/s running. Top: Render of single rigid-body model optimized trajectory. Middle: Cassie MuJoCo simulation. Bottom: Cassie hardware treadmill test.

itself as well as the limitations of its application to Cassie. Firstly, the model only has point feet and no swing leg dynamics, to account for these we reward foot orientation and swing apex heights to encourage the system to keep its feet facing forward and for the robot to not shuffle its feet.

$$r_{\mathbf{q}} = 0.3\ \exp(-|d(\mathbf{q}_{\text{foot}}, \mathbf{q}_{\text{foot}}^{\text{ref}})|) \tag{8.17}$$

$$r_{z_{\text{foot}}} = 0.3\ \exp(-|z_{\text{foot}} - z_{\text{foot}}^{\text{des}}|). \tag{8.18}$$

A major point of consideration is how to impart single rigid-body orientations onto a multiple-rigid body complex system. Geometric mechanics techniques such as minimum perturbation coordinates [153] possibly present intriguing insights but are out of scope for this work. To this effect we chose to match our orientations to Cassie's pelvis, which only represents a portion of the robots total orientation, but is the single most natural choice of its many rigid bodies. As the pelvis is not the optimal choice to apply the SRBM rotational information we found it

Parameter	Randomization Bounds
Policy Rate	[0.95:1.05] $\times$ default
Joint Damping	[0.5:3.5] $\times$ default
Joint Mass	[0.5:1.5] $\times$ default
Ground Friction	[0.35:1.1]

Table 8.2: Dynamics randomization parameter ranges.

necessary to include terms to our reward formulation that attempt to ameliorate this discrepancy. A lateral drift term is included to keep the robot running in a straight line,

$$r_{\text{drift}} = \begin{cases} 0.3 & \text{if } |p_y| < 0.2 \\ 0.3 \exp(-|15 \cdot p_y|) & \text{if } |p_y| \geq 0.2 \ . \end{cases} \tag{8.19}$$

The final rewards are penalties to hip roll motor velocities to prevent excessive motions at the pelvis.

$$r_{\text{hip roll}} = 0.1 \exp(-|\omega_{\text{hip roll}}|) \tag{8.20}$$

$$r_{\text{hip yaw}} = 0.1 \exp(-|\omega_{\text{hip yaw}}|). \tag{8.21}$$

All reward terms are then normalized before summing to calculate the total reward. In order to increase our policies robustness and aid in the crossing of the sim-to-real gap we employ dynamics randomization during training. At each episode the dynamics listed in Table 8.2 are randomly modified to aid in the generalization of our policy and its ability to overcome discrepancies from simulation to hardware:

8.7 Running Results

Using our dynamic maneuver TO formulation outlined in Section 8.5 we are able to create a wide variety of reference maneuvers based on SRBM dynamics as applied to the Cassie robot. Several such example maneuvers and their ground reaction forces are presented in Fig. 8.4 and in the supplemental video.

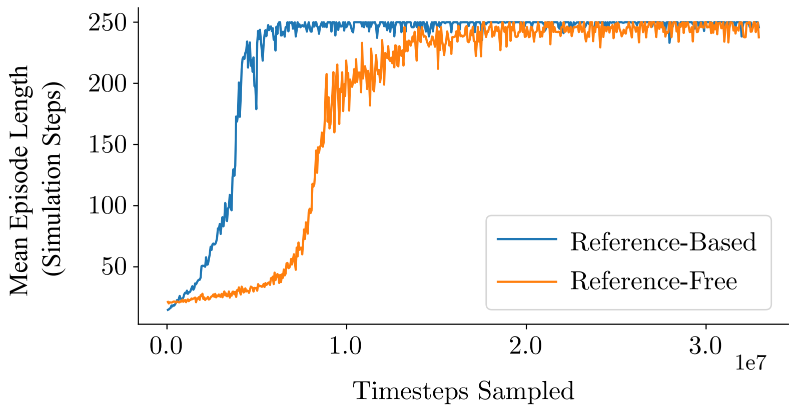


Figure 8.6: Mean training episode length for grounded running trained on speeds between 0.1 and 3.5 m/s. The episode length is an useful way to compare ability of controllers with different reward functions.

8.7.1 Simulation Results

Two running policies were trained with the same inputs and hyperparameters over the velocity range of 0.1-3.5 m/s. First is our reference-based control policy with the same composition as outlined in Section 8.6. The second serves as a reference-free baseline and utilized the previous state of the art reward composition laid out in [141]. Given the differences in reward terms, we elect to compare mean episode length for each policy during training as a metric for quantifying the stability of each policies gait, as shown in Fig. 8.6. We can note that the reference-based policy achieves a stable running gait (consistently reaches the maximum episode length of 250 simulation steps) with much greater sample efficiency than the reference-free policy. A stable running gait is achieved by the reference-based policy after only $\sim 0.8\times10^7$ timesteps where the reference-free policy required $\sim 2.0\times10^7$ timesteps, indicating that for this particular gait the usage of SRBM reference dynamics yielded a roughly 2.5$\times$ increase in sample efficiency. From qualitatively analyzing both policies in simulation, we note that the reference-based policy moves in a more fluid manner, characterized by the natural oscillating motions of the model when compared to the more rigid reference-free policy. This can be seen in the supplemental video.

8.7.2 Hardware Results

The reference-based control policy was successfully deployed onto a physical Cassie robot and was able to run on a treadmill up to 3.0 m/s as shown in the supplemental video. A high level of similarity can be visually seen between the gaits of the SRBM reference (with added swing leg kinematics for visualization), the MuJoCo simulation in which training took place, and the physical hardware test, as shown in Fig. 8.5. This demonstration indicates that SRBM dynamic reference information, even with rotational dynamics applied only at Cassie's pelvis can indeed be used in a RL framework to train dynamic running policies up to high speeds. However, we were not able to fully cross the sim-to-real gap as top speeds (between 3.0 and 3.5 m/s) achieved in simulation were not able to be repeated on hardware.

8.8 Four-Step Reinforcement Learning Problem

A four step 90 right turn was selected as the target behavior for the control policies in this work because its aperiodic and highly dynamic nature marks a significant departure from the dynamical regime of regular walking. Control policies trained to execute the turning behavior are initialized from states derived from a pretrained walking policy and transition back to the walking policy at the end of the turning maneuver. Matching the terminal state of the reference trajectory is not guaranteed to permit successful transitions back to walking policies, so turning policies must learn how to deviate away from tracking the reference data to facilitate successful transitions. This challenge is addressed using epilogue rewards detailed in Section 8.8.5, and is a novel component of our proposed learning framework. We train recurrent control policies to perform four-step, 90 turns using Proximal Policy Optimization (PPO) in simulation [136]. The simulator we use is MuJoCo, extended with the robot's state estimator and noise models[2]. Previous work has shown that highly accurate simulations such as this are effective at producing control policies that transfer to hardware with no additional adaptation [40, 141, 63].

[2]Simulation available at https://github.com/osudrl/cassie-mujoco-sim

8.8.1 Reference Trajectory Optimization

Dynamic legged maneuvers require abrupt changes in linear and angular momentum while heavily constrained by underactuation constraints. We hypothesize that in these contexts reference information could be more useful than it was previously shown to be in nominal, steady-state locomotion. To provide a rich library of reference motions we perform trajectory optimization with an SRBM, representing a reduced-order model of locomotion. The SRBM approximates the complex multibody dynamics of a robot into a single rigid-body with dynamics that are manipulated via ground reaction forces applied at footholds. We apply a widely-used, prescribed contact sequence, direct collocation trajectory optimization method [84]. This allows the optimization to adjust foot timings, but not the sequence of contacts. This is not overly restrictive as bipedal robots have only a small space of feasible contact patterns. Our contact pattern for four-step turns is a grounded run consisting of alternating phases of single-stance with instant transfer. We apply a set of transferability constraints which ensure the resulting trajectories are more directly applicable to the target Cassie robot. These include maximum ground reaction force, friction cones, maximum yank (time rate of change of force), leg length limits, and foot placement constraints to prevent leg crossing. More details on the library generation method can be found in [13].

The resulting library of turn references spans from 0.0 to 2.5 m/s. The 2.5 m/s turning trajectory is shown in Fig. 8.7. The trajectories have smoothly varying body motions, footstep locations, ground reaction forces, and step timing.

8.8.2 Policy Network Design

The policy architecture used in this work is derived from previous work on applying LSTM networks to bipedal locomotion control [143]. Both actor and critic networks are LSTM RNNs of size 128x128. The state space inputs to our control policy concatenates information from the robot state estimator along with a maneuver progression counter, two periodic clock waveforms, and a target forward heading

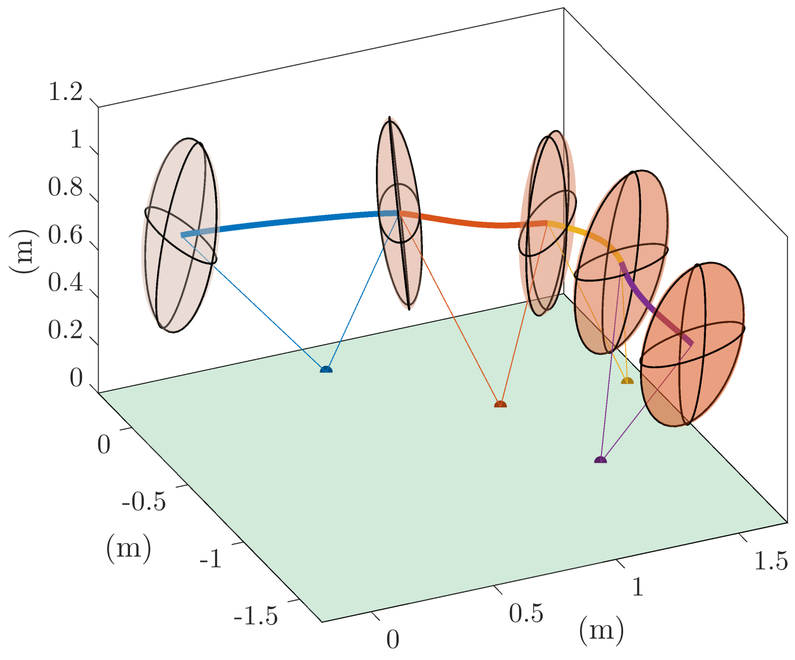


Figure 8.7: Plot of the reference trajectory for a 2.5 m/s, four-step turn from the optimized single rigid-body model moving left to right. The thick line represents the center of mass path, with different colors showing the different stance phases. Thin lines show leg positions at the start and end of stance phases.

Policy Input	**Size**
Pelvis Orientation Quaternion	4
Pelvis Angular Velocity	3
Pelvis Translational Acceleration	3
Joint Positions and Velocities	28
Maneuver Progression	1
Clock Signal	2
Target Forward Speed	1

Table 8.3: The inputs into the learned control policies. All state information is estimated from real or simulated sensor data.

speed for a total input space size of 42. The breakdown of the state space is shown in Table 8.3. From our testing, using a reduced state input set by omitting the maneuver progression counter and pelvis translational acceleration estimates also produces successful turning policies with no noticeable difference in simulation behavior and training time.

The action space of the policy consists of position targets for all 10 actuated joints on Cassie. The actions are updated at our nominal policy control rate of 40 Hz, which are then sent to joint-level PD controllers running at 2 kHz.

8.8.3 Reward Function Formulation

To support an ablation study, we trained policies using different reward functions, *Full Reference, Subset Reference, Foot Timing,* and *No Reference*, each with a different set of additive reward components that capture different aspects of reference information. Table 8.4 gives the individual component weights for each of the four reward functions. All weights are rounded the nearest percentage point.

Two of the reward components are common across all four reward function variations:

- A contact mode reward r_{contact}, which specifies when each foot should be in swing or in stance with a piecewise linear clock function. The gait parameters that define such a function (stepping frequency and swing ratio) and calculated from the reference information. We refer readers to previous work

[40] for further details.

- Action smoothness, torque cost, motor velocity costs r_{ctrl} on the hip roll and yaw motors, and self collision avoidance rewards to promote successful sim-to-real transfer.

The four reward functions are summarized below.

8.8.3.1 Full Reference

The tracking components of the reward function include pelvis yaw angle (ψ), pelvis linear velocity ($\mathbf{v}$), pelvis angular momentum ($\mathbf{L}$), and the relative distance vector between the pelvis COM and the stance foot (**pose**). They are given by

$$r_{\psi}^{\text{ref}} = \exp\left(-\left|(3(\psi_{\text{pelv}} - \psi_{\text{pelv}}^{\text{ref}})\right|\right) \tag{8.22}$$

$$r_{\mathbf{v}} = \exp\left(-\left\|2(\mathbf{v}_{\text{pelv}} - \mathbf{v}_{\text{pelv}}^{\text{ref}})\right\|_1\right) \tag{8.23}$$

$$r_{\mathbf{L}} = \exp\left(-\left\|\mathbf{L}_{\text{body}} - \mathbf{L}_{\text{body}}^{\text{ref}}\right\|_1\right) \tag{8.24}$$

$$r_{\mathbf{pose}} = \exp\left(-\left\|5(\mathbf{p}_{\text{pose}} - \mathbf{p}_{\text{pose}}^{\text{ref}})\right\|_1\right) \tag{8.25}$$

8.8.3.2 Subset Reference

Includes only a subset of full reference rewards, specifically $r_{\psi}^{\text{ref}}, r_{\mathbf{v_{xy}}}, r_{\mathbf{pose_{xy}}}, r_{\text{contact}}, r_{\text{ctrl}}$. Notably, this omits the angular momentum tracking term in equation (8.24), as well as tracking only the planar x, y components of $r_{\mathbf{v}}$ and $r_{\mathbf{pose}}$. This particular reward function was chosen because angular momentum tracking was found to have no noticeable effects on the behavior of the resulting policies.

Reward	**Full Ref.**	**Sub Ref.**	**Foot Timing**	**No Ref.**
r_{ψ}^{ref}	6	3	-	-
r_{ψ}^{interp}	-	-	20	20
$r_{\mathbf{v_{xy}}}$	6	22	-	-
$r_{\mathbf{v_z}}$	3	-	-	-
$r_{\mathbf{pose_{xy}}}$	13	22	-	-
$r_{\mathbf{pose_z}}$	6	-	-	-
$r_{\mathbf{L}}$	9	-	-	-
r_{contact}	38	32	53	53
r_{ctrl}	19	22	27	27

Table 8.4: Reward component composition and weighting percentages.

8.8.3.3 Foot Timing

Omits all tracking rewards (8.22) to (8.25) and only consists of $r_{\psi}^{\text{interp}}, r_{\text{contact}}, r_{\text{ctrl}}$. The only reference information present in the reward is the gait parameters for the contact mode reward term. r_{ψ}^{interp} replaces r_{ψ}^{ref} and tracks a yaw target that linearly interpolates between 0 and $-\pi/2$ within the timespan of the reference turning maneuvers instead of the optimized yaw trajectory $\psi_{\text{pelv}}^{\text{ref}}$.

8.8.3.4 No Reference

A reference-free policy similar to *Foot Timing* that also omits the tracking rewards (8.22) to (8.25). It uses $r_{\psi}^{\text{interp}}, r_{\text{contact}}, r_{\text{ctrl}}$ exclusively, but in contrast to *Foot Timing*, the gait parameters for r_{contact} are set by a hand-tuned heuristic. Thus, this policy uses **no** information from the reference trajectory.

8.8.4 Episode Initialization

The beginning of a training episode for turning needs to be reset to a configuration that is a close match to the starting SRBM configuration specified by the turning trajectory. For this purpose, a set of initialization poses $P_{\text{init}}(v, \theta)$ is generated by executing a pre-trained running policy in simulation for a sweep of commanded speeds that match the speeds v of the trajectories in the reference library. The configurations $[q, \dot{q}]$ of Cassie within a range of gait phases θ before and after a left-foot swing apex (the starting point of the reference trajectories) are saved to P_{init}. On every reset, the configuration state of Cassie is uniformly sampled from the set of poses in P_{init} for a desired initial speed.

8.8.5 Epilogue Reward

Since we want to transition back to walking after executing a turn, it is paramount that the turning policy fulfill the terminal objective of ending in a state that can successfully initialize walking in order to return to a nominal locomotion gait. We introduce the novel concept of training with an epilogue reward as a component of our training framework in order to allow turning policies π^{turn} to successfully switch back to the nominal locomotion policies π^{walk} once the turning policies have reached the end of the reference trajectory states.

At the end of a PPO rollout, the critic value for the final state $V_{\pi}(S_T)$ is used as the terminal value in the discounted chain of rewards received at each episode step to estimate the sum of any future discounted rewards [136]. This is analogous to calculating the n-step temporal-difference (TD) returns, where the final value is an estimate for the uncollected rewards beyond the n-step horizon [148].

The epilogue reward introduced in this work modifies the terminal value used at the end of a π^{turn} training episode. It is computed as the discounted sum of returns of the epilogue episode, which triggers when the turning policy has successfully reached the end of its maneuver. In the epilogue episode, the walking policy π^{walk} takes over from the last state of the turning episode (normal PPO rollout of the

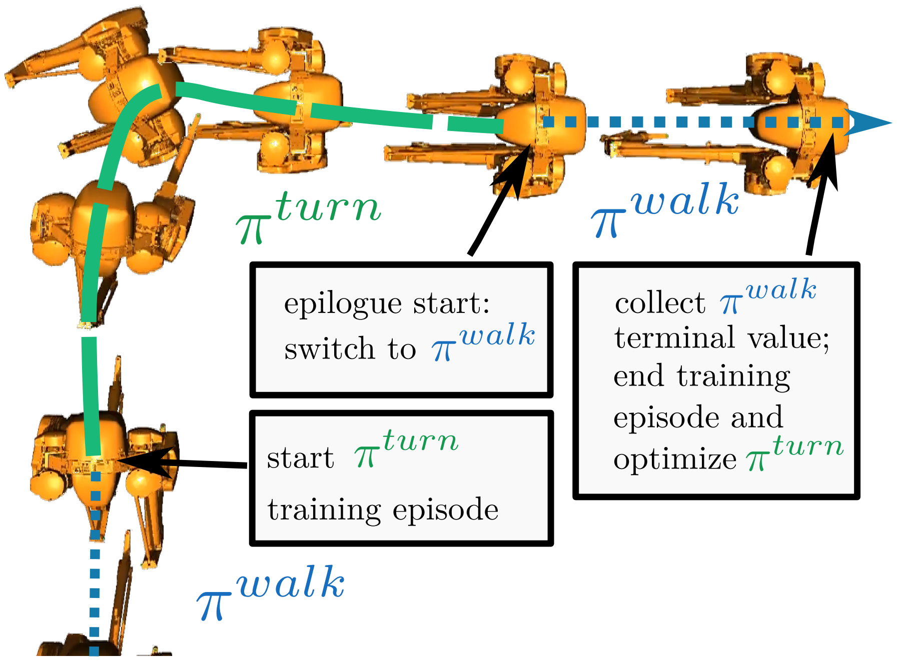

π^{turn}
π^{walk}
epilogue start: switch to π^{walk}
collect π^{walk} terminal value; end training episode and optimize π^{turn}
start π^{turn} training episode
π^{walk}

Figure 8.8: Visualization of a PPO rollout during training. After being initialized from a π^{walk} pose, π^{turn} is evaluated until the end of the turning maneuver. If π^{turn} completed the turning maneuver, π^{walk} subsequently takes over to generate the epilogue reward.

π^{turn} turning policy), and is evaluated deterministically for k simulation steps. Once the epilogue is complete, the epilogue reward becomes the value of $V_{\pi^{\text{turn}}}(S_T)$ used to optimize π^{turn}. Formally, the epilogue reward is

$$V_{\pi^{\text{turn}}}(S_T) = \Big(\sum_{i=T}^{T+k} \gamma^{i-T} R_{i+1}\Big) + \gamma^{T+k} V_{\pi^{\text{walk}}}(S_{T+k+1}) \tag{8.26}$$

where k is the length of the epilogue, T is the length of the turning maneuver, R_{i+1} is based on the reward function used to train π^{walk}, and $V_{\pi^{\text{walk}}}$ is the critic trained for the walking policy. Modifying the estimate of future returns in this manner incentivizes π^{turn} to terminate in a configuration $[q, \dot{q}]$ amenable for the execution of π^{walk} by maximizing the epilogue returns for continued walking. As a control for the epilogue, a *No Epilogue* policy is trained using the exact same rewards as *Foot Timing*, but with k set to 0. The *Foot Timing* reward set was chosen over the others because it produced the highest quality turning behaviors in sim as well as fast convergence times. A value of 120 is used for k for all other policies.

8.8.6 Dynamics Randomization

We applied dynamics randomization as described in [143] during the training process of our turning controller to help close the sim-to-real gap and enable a successful transfer to real hardware. In addition, we also apply a constant perturbance force to the robot pelvis over the course of a training episode with a randomly sampled magnitude and direction in order to promote the emergence of robust turning behaviors. The details of our randomization parameters can be found in Table 8.5.

8.9 Four-Step Turn Results

To evaluate the utility and necessity of our optimized SRBM trajectories and the epilogue reward, we assess and compare the set of policies proposed in Section 8.8.3

Parameter	**Range**	**Unit**
Policy Control Rate	[0.95,1.05] $\times$ default	Hz
Joint Encoder Noise	[-0.05, 0.05]	rad
Joint Damping	[0.8, 2.5] $\times$ default	Nms/rad
Link Mass	[0.9, 1.5] $\times$ default	kg
Friction Coefficient	[0.45, 1.3]	-
External Force Magnitude	[0, 40]	N
External Force Dir. (Azimuth)	[0, 2π]	rad
External Force Dir. (Elevation)	[0, $\frac{\pi}{4}$]	rad
Initial Pelvis Velocity (x)	[-0.3,0.3] + default	m/s
Initial Pelvis Velocity (y)	[-0.4,0.4]	m/s

Table 8.5: Dynamics Randomization Parameter Range

and Section 8.8.5 in simulation for their performance and turning behavior characteristics. We also present successful sim-to-real transfer of a selection of the policies tested in simulation in our submission video.

8.9.1 Simulation Results

8.9.1.1 Sample Efficiency

We plot the learning curves for each policy in Figure Fig. 8.9 to compare the sample efficiency of our turning policies. Since each policy is trained with different reward functions, the reward values attained by each policy can not be used to form conclusions about their relative performance. Instead, we compare policies by the number of samples to convergence, shown for each turning policy by star symbols in Fig. 8.9 that mark when each policy first surpassed 97% of the maximum reward value experienced during training. Notably, the policies that use less information from the optimized trajectories converge slightly faster than the policies that follow the SRBM reference trajectory more faithfully. We hypothesize that the learning

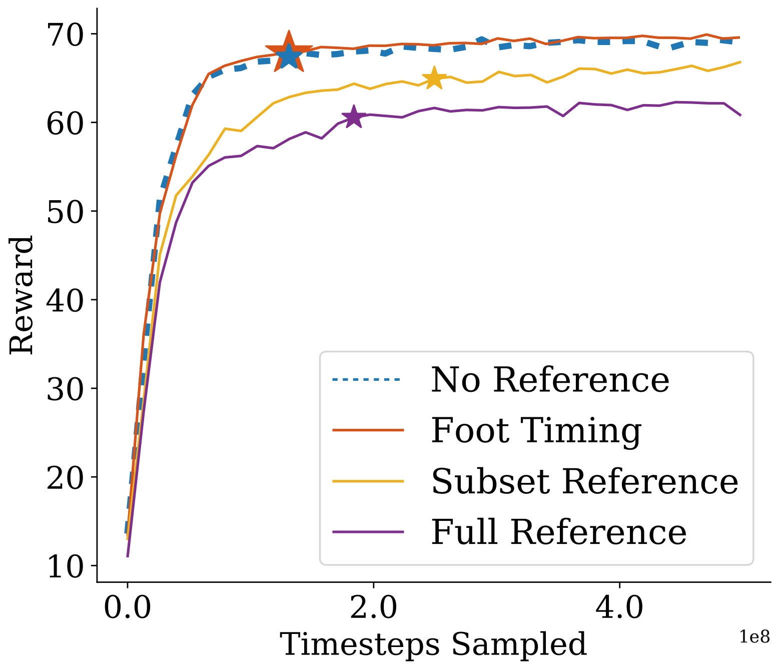


Figure 8.9: Comparison of sample efficiency for our proposed turning policies. Note that the absolute scale of the different curves are not necessarily comparable since each reward function include different reward components. The star symbols mark the time to convergence for each policy, which is the point on the learning curve that exceeds 97% of the maximum reward seen during training for the first time.

speed disparity may be attributed to model differences between the SRBM and Cassie's full-order dynamics leading to conflicting interactions between the tracking reward terms. This may cause policies that track more of the reference data to require more samples in order to learn to optimize for multiple conflicting objectives before convergence.

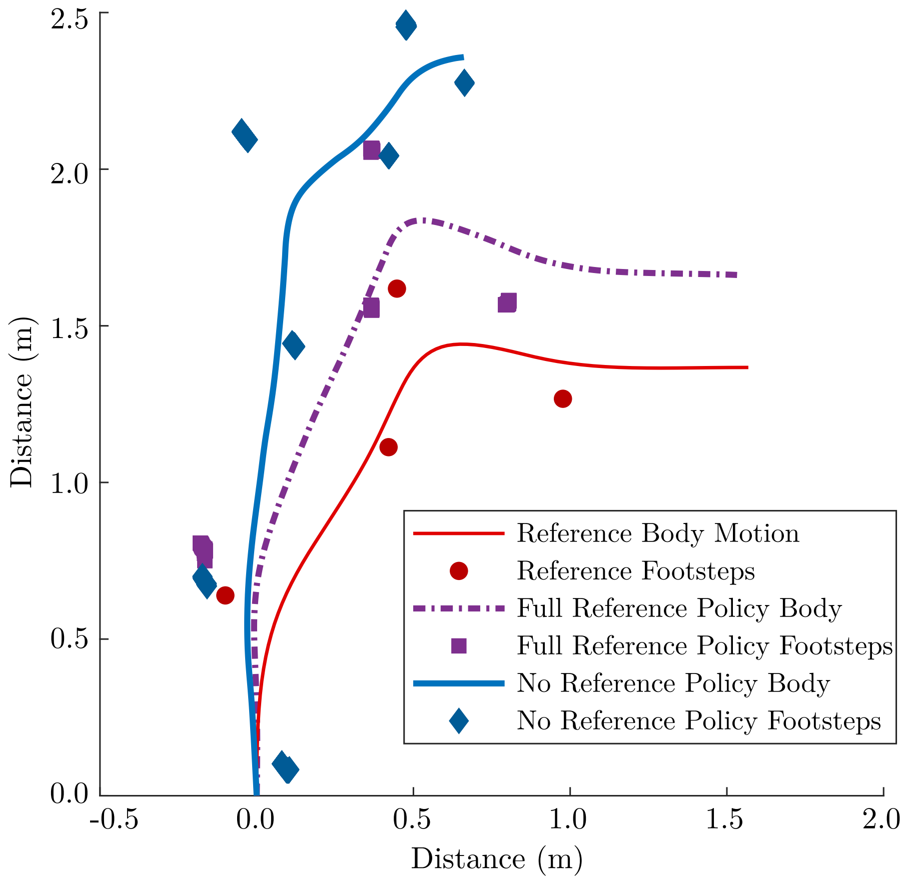


Figure 8.10: Plot of footstep touchdown locations and pelvis trajectory for the reference data, *Full Reference* and *No Reference* policies for a turning maneuver executed at 2.5m/s.

8.9.1.2 Turning Behavior

Fig. 8.10 compares the turning trajectory of the *Full Reference* and *No Reference* policies for a single sample trial in simulation against the trajectory prescribed by the reference data. Since the *No Reference* policy is trained to match a footstep contact schedule set by a heuristic instead of following the trajectory data, it completes the 90 turn in seven steps rather than four. As a result, the pelvis trajectory and footstep placements distinctly differs from the reference data since it is trained to not track the reference data. This is in contrast to the *Full Reference* policy turning behavior, where the features of the pelvis trajectory is similar to that of the reference data, and the placement of its stance feet relative to the body also closely match those of the reference. Policies trained on subsets of the tracking rewards all produce four-step turning behaviors similar to the results of the *Full Reference* policy, indicating that the only necessary reference trajectory information for training policies to perform four-step 90 turns is a feasible footstep contact schedule. Fig. 8.11 illustrates the change in orientation of the robot pelvis over the course of a turning maneuver, which is not communicated by the pelvis COM trajectories illustrated in Fig. 8.10. The *Full Reference* and *Subset Reference* policies are the two policies rewarded to track the optimized body yaw angle trajectory, but we see observe noticeable deviations from the target yaw trajectories at the beginning of the first and third footsteps. This is likely caused by the policies learning to maximize rewards of multiple conflicting objectives from the reference data, such as pelvis linear velocity and body yaw angles.

8.9.1.3 Policy Robustness

We simulate 1000 trials of 2.5m/s turning maneuvers for each turning policy to assess its ability to complete a turn and switch back to π^{walk} successfully in the presence of a constant perturbance force applied to the body during the execution of the turning maneuver. A random direction is sampled before each turning maneuver trial, and a constant force of 35N is applied in the chosen direction.

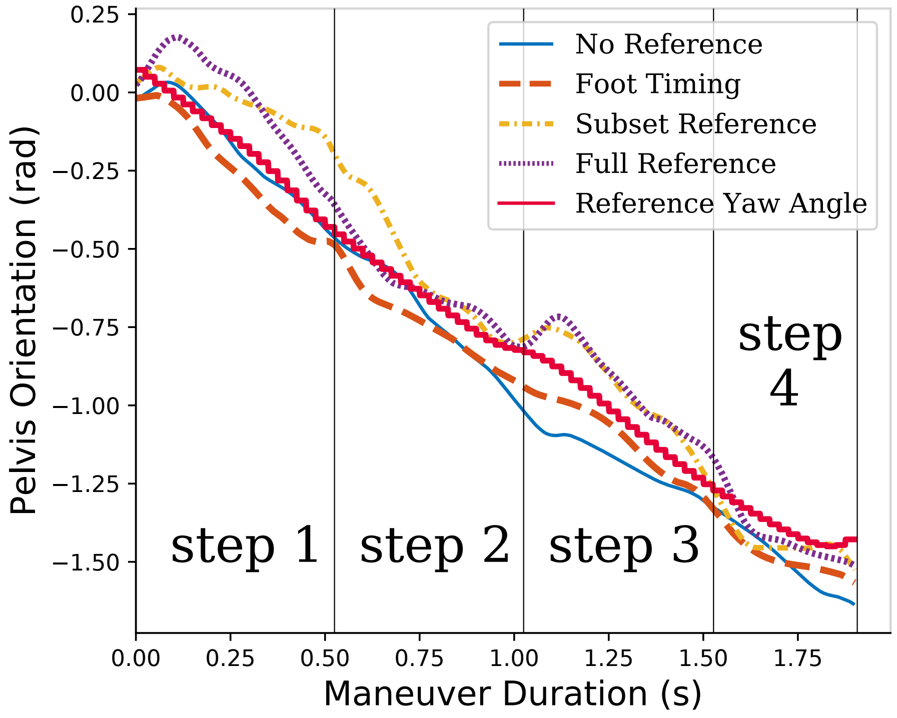


Figure 8.11: Simulated pelvis yaw angles for various turning policies over the course of a turning maneuver. The Foot Timing and No Reference policies are not trained trained to track the reference data yaw trajectory shown in solid red.

The time at which the the policy falls over is logged for each trial to produce the policy survival plots shown in Figs. 8.12 and 8.13. From Fig. 8.13, policies trained with more of the reference trajectory data are more robust at rejecting perturbance forces, with the exception that the *No Reference* policy outperforms the *Foot Timing* policy during the turn maneuver. Since all policies experience perturbance forces during training, it is possible that rewarding policies to track certain elements of the trajectory data such as foot placement positions and pelvis translational velocities are beneficial to policy robustness. Fig. 8.12 compares the effects of training with and without the use of epilogue rewards. While the *Foot Timing* policy achieved a survival fraction of around 50%, the *No Epilogue* policy fared significantly worse than its counterpart with a terminal survival fraction of just 4%. We observed that *No Epilogue* policy had a low transition success rate, which indicates that epilogue rewards are necessary for reliable switching between π^{turn} and π^{walk} policies. We hypothesize that using other reward functions without the epilogue reward will produce similar results, but did not run such tests in this work.

8.9.2 Hardware Results

During our outdoors hardware tests, we were able to demonstrate successful turning maneuvers and policy switching on artificial turf with the *No Reference* and *Full Reference* policies. The *Subset Reference* policy was also tested, but we were unable to switch back to the walking policy without falling. We also successfully tested the *Foot Timing* policy indoors on multiple low speed turns performed in succession. While our simulation results indicate that the *Full Reference* policy should perform more consistently than the *No Reference* policy on real hardware, we observed that the *No Reference* policy was more consistent than the *Full Reference* policy at turning and transitioning in our outdoors tests. The *Full Reference* and *Subset Reference* policies pitch the pelvis during the turn more so than the *No Reference* and *Foot Timing* policies that in contrast keep the pelvis fairly level throughout the turn. This is consistent with what we see in simulation, although

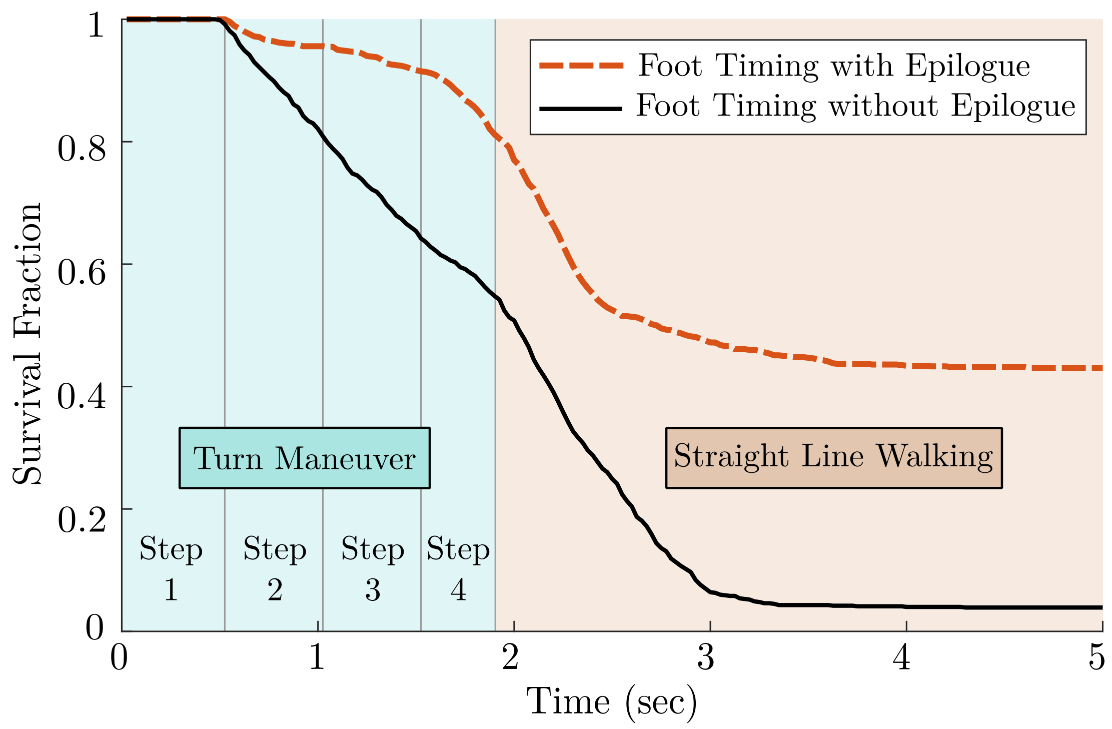


Figure 8.12: Robustness comparison between the *Foot Timing* policy trained with epilogue and the *Foot Timing* policy trained without using the same method as Fig. 8.13.

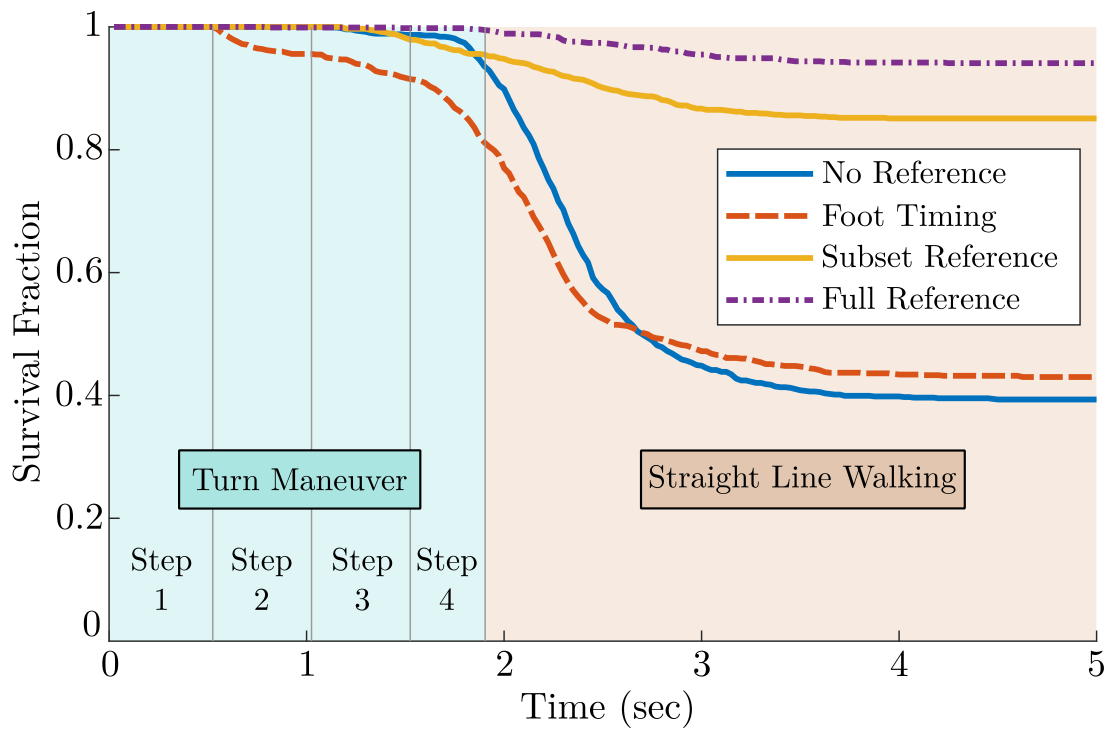


Figure 8.13: Robustness comparison between our proposed turning policies conducted for 1000 turning maneuver trials at 2.5m/s. The step labels denote the reference data step progression timings produced by TO. Since the *No Reference* policy is trained to follow a contact schedule set by a heuristic, the step labels do not apply to this policy.

the *Full Reference* pelvis pitching motion in simulation is more fluid and intricate than our corresponding hardware results which maintain an awkward downward pitch throughout the entire turn. Due to unresolved sim-to-real challenges, we were unable to replicate the same consistent performance seen in simulation in our hardware trials. From our limited hardware tests, the *No Reference* policy seemed to have a higher chance at executing successful turns than the *Full Reference* policy. We refer readers to the attached video for full hardware results.

8.10 Conclusion

In this work we presented a framework for both authoring and executing dynamic maneuvers for bipedal robots. Using a single rigid-body model-based trajectory optimization amended for bipeds, we formulated a method of developing reference trajectories for arbitrary maneuvers by specifying sequences of hybrid modes with sets of shaping constraints. We were also able to demonstrate that these reference trajectories could be used to develop reinforcement learning control policies capable of crossing the sim-to-real gap to actual hardware. Our proposed method yielded a $\sim 2.5\times$ increase in training sample efficiency to stable locomotion and our deployed policy was able to achieve speeds of 3.0 m/s on hardware.

Epilogue rewards are a key component of our proposed learning framework to facilitate smooth transitions between nominal locomotion policies and the policy trained to execute the desired maneuver. While our methods exhibited promising results in simulation, we encountered difficulties with sim-to-real and were unable to fully transfer the success of our simulation results to hardware field trials. Although we were able to demonstrate repeated successful turning maneuvers at low speeds on hardware, turning maneuvers committed at faster walking speeds were much less reliable. A possible reason for the performance gap might stem from issues with the hardware state estimator producing inaccurate orientation estimates only when large pelvis accelerations are experienced during the execution of turning maneuvers at higher speeds. Our reference-based policies command much larger pelvis pitch angles over the course of a turn than the *No Reference* and *Foot Timing* policies which may have exacerbated the state estimation problems. It is possible that this difference may be a contributing factor to why the *No Reference* policy performed better than the reference-based policies on hardware when our simulation results suggest the opposite. One downside of the learning framework introduced in this work is the challenge of scaling to switching between multiple aperiodic behaviors sourced from a diverse behavior trajectory library. Future avenues of research could build upon this work by investigating how to effectively switch between large sets of individual behavior policies in order to allow for the execution of more complex dynamic routines such as dancing or parkour.

Chapter 9: General Conclusions

This thesis covered many techniques, approaches, and contexts in its pursuit of agile legged locomotion. Each chapter contained discussion and conclusions relevant to its own content. This chapter will expand on what I listed as most impactful contributions from this thesis and explicitly describe how they relate back to the fundamental understanding of agile legged locomotion articulated in Part I.

Specific evidence that swing leg extension and angular retraction produce a more robust response to ground variations. This work builds the body of work which shows that open loop swing leg motions are able to stabalize running in the presence of ground disturbances. In particular this builds on previous work with passive models in the context of guinea fowl running [21] and preflexes [156] by looking at open loop actuation plans which are not replanned at all during stance. The results further show that the combination of open loop strategies and leg passive dynamics can fully reject ground disturbances without active reactions or replanning. This directly relates to the concept that designing behaviors around the moment of touchdown can produce motions that are more robust to disturbances.

An approach to mitigate the risk of invalid solutions when planning with data-driven Poincarè models. This supports the idea that high level planning decisions, like footstep placement, can be made in a reduced space compared to the full order system. In the work, we explicitly only plan using the surface of section which orbits pass through. It also examines some of the ideas of trading off between aversion to falls and highly dynamic motions through the tuning of the novel failure margin function threshold.

The first demonstration of blind traversal of stairs by an unsupported bipedal robot. Previous work has looked as precise sensing and planning to climb and descend flights of stairs but this is the first to successfully demonstrate this without sensing or modeling the stairs. It was highly successful, being able to climb up and down a half flight of stairs ten times back-to-back with only one failure.

Evidence that learning locomotion on stair environments resulted in increased swing leg retraction compared to flat ground environments. This reinforces that careful consideration of the moment of touchdown is vital to be robust to ground variations. It further relates to biological studies of animal swing leg motions by closely matching their reactions to drop steps.

The first demonstration of training a learned controller for a biped to emulate motions from a reduced-order model. This reinforced the assessment that locomotion exists on a reduced manifold by creating walking and running gaits for a reduced order model then using reinforcement learning to elevate those motions to the full order robot. It specifically supports this because the final motion of the robot is identifiably similar to the spring mass model's motion. The body motion is perceptively similar and the ground reaction forces match the double humped shape from the model (without explicit reward to match force profiles).

The first example of learned, high-speed, agile turning maneuvers on an unsupported bipedal robot. The extends much of the previous work in this thesis and the field by pushing learned behaviors beyond the space of nominally steady state locomotion. Similar behaviors have been demonstrated by Boston Dynamics on the Atlas robot, however specific details on their methods are limited and from the information they have provided it appears their methods are substantially different than the approach used in this work [22].

A novel epilogue reward system for learning fixed duration maneuvers. This is particularly important for because of learned robotic control methods typically use long, discounted reward horizons to incentivize stability. When a maneuver has an end, there needs to be a strong reward signal to ensure this final state is a good state for the next controller to start from. The new epilogue reward system is vital for creating a maneuver controller which can transition back to walking or running.

Bibliography

[1] Mini cheetah is the first four-legged robot to do a backflip. `http://news.mit.edu/2019/mit-mini-cheetah-first-four-legged-robot-to-backflip-0304`. Accessed: 2019-04-05.

[2] Andrew M. Abate. *Mechanical Design for Robot Locomotion*. PhD thesis, Oregon State University, 2018.

[3] Farzad Adbolhosseini, Hung Yu Ling, Zhaoming Xie, Xue Bin Peng, and Michiel van de Panne. On Learning Symmetric Locomotion. In *Proc. ACM SIGGRAPH Motion, Interaction, and Games (MIG 2019)*, 2019.

[4] M. Ahmadi and M. Buehler. The ARL monopod II running robot: control and energetics. In *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, volume 3, pages 1689–1694. IEEE.

[5] Junhyeok Ahn, Donghyun Kim, Seunghyeon Bang, Nick Paine, and Luis Sentis. Control of a high performance bipedal robot using viscoelastic liquid cooled actuators. *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, 10 2019.

[6] Amos Albert, Michael Suppa, and Wilfried Gerth. Detection of stair dimensions for the path planning of a bipedal robot. In *2001 IEEE/ASME International Conference on Advanced Intelligent Mechatronics. Proceedings (Cat. No. 01TH8556)*, volume 2, pages 1291–1296. IEEE, 2001.

[7] R McNeill Alexander. *Elastic mechanisms in animal movement*. Cambridge University Press, 1988.

[8] Richard Altendorfer, Daniel E Koditschek, and Philip Holmes. Stability analysis of a clock-driven rigid-body slip model for rhex. *The International Journal of Robotics Research*, 23(10-11):1001–1012, 2004.

[9] Jonathan Amos. Uk drones map chernobyl's 'red forest'. *BBC News*, May 2019.

[10] ANYbotics. Meet anymal, your new inspector. "`https://www.anybotics.com/anymal-autonomous-legged-robot/`", 2021.

[11] Taylor Apgar, Patrick Clary, Kevin Green, Alan Fern, and Jonathan Hurst. Fast online trajectory optimization for the bipedal robot cassie. In *Proc. Robotics: Science and Systems XIV*, Pittsburgh, PA, USA, 2018.

[12] Apptronik. Draco biped. "`https://www.anybotics.com/anymal-autonomous-legged-robot/`", 2021.

[13] Ryan Batke, Fangzhou Yu, Jeremy Dao, Jonathan Hurst, Ross L. Hatton, Alan Fern, and Kevin Green. Optimizing bipedal maneuvers of single rigid-body models for reinforcement learning, 2022.

[14] Klaus-Peter Beier and Yifan Chen. Highlight-line algorithm for realtime surface-quality assessment. *Computer-Aided Design*, 26(4):268–277, 1994. Special Issue: Mathematical methods for CAD.

[15] Pranav A Bhounsule, Jason Cortell, and Andy Ruina. Design and control of ranger: an energy-efficient, dynamic walking robot. In *Adaptive Mobile Robotics*, pages 441–448. World Scientific, 2012.

[16] Pranav A. Bhounsule, Myunghee Kim, and Adel Alaeddini. Approximation of the Step-to-Step Dynamics Enables Computationally Efficient and Fast Optimal Control of Legged Robots. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 10: 44th Mechanisms and Robotics Conference (MR), 08 2020.

[17] A. V. Birn-Jeffery, C. M. Hubicki, Y. Blum, D. Renjewski, J. W. Hurst, and M. A. Daley. Don't break a leg: running birds from quail to ostrich prioritise leg safety and economy on uneven terrain. *Journal of Experimental Biology*, 217(21):3786–3796, 2014.

[18] Gerardo Bledt, Matthew J Powell, Benjamin Katz, Jared Di Carlo, Patrick M Wensing, and Sangbae Kim. MIT Cheetah 3: Design and control of a robust, dynamic quadruped robot. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2245–2252. IEEE, 2018.

[19] Michael Bloesch, Marco Hutter, Mark Hoepflinger, Stefan Leutenegger, Christian Gehring, C. David Remy, and Roland Siegwart. State Estimation for Legged Robots - Consistent Fusion of Leg Kinematics and. *Robotics: Science and Systems VIII*, 2012.

[20] Yvonne Blum, Susanne W Lipfert, Juergen Rummel, and André Seyfarth. Swing leg control in human running. *Bioinspiration & biomimetics*, 5(2):026006, 2010.

[21] Yvonne Blum, Hamid R. Vejdani, Aleksandra V. Birn-Jeffery, Christian M. Hubicki, Jonathan W. Hurst, and Monica A. Daley. Swing-leg trajectory of running guinea fowl suggests task-level priority of force regulation rather than disturbance rejection. *PLoS ONE*, 9(6):18–20, 2014.

[22] Boston Dynamics. Atlas — partners in parkour, 2021.

[23] Philemon Brakel, Steven Bohez, Leonard Hasenclever, Nicolas Heess, and Konstantinos Bousmalis. Learning coordinated terrain-adaptive locomotion by imitating a centroidal dynamics planner. *CoRR*, abs/2111.00262, 2021.

[24] B. Brown and G. Zeglin. The bow leg hopping robot. In *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*, volume 1, pages 781–786. IEEE, 1998.

[25] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J.J. Leonard. Past, present, and future of simultaneous localization and mapping: Towards the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332, 2016.

[26] Stéphane Caron, Abderrahmane Kheddar, and Olivier Tempier. Stair climbing stabilization of the HRP-4 humanoid robot using whole-body admittance control. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 277–283. IEEE, 2019.

[27] Guillermo A Castillo, Bowen Weng, Wei Zhang, and Ayonga Hereid. Robust feedback motion policy design using reinforcement learning on a 3d digit bipedal robot. *arXiv preprint arXiv:2103.15309*, 2021.

[28] Yu-Ming Chen and Michael Posa. Optimal reduced-order modeling of bipedal locomotion. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8753–8760, 2020.

[29] Christine Chevallereau, Gabriel Abba, Yannick Aoustin, Franck Plestan, Eric Westervelt, Carlos Canudas De Wit, and Jessy Grizzle. Rabbit: A testbed for advanced control theory. *IEEE Control Systems Magazine*, 23(5):57–79, 2003.

[30] Patrick Clary, Pedro Morais, Alan Fern, and Jonathan Hurst. Monte-carlo planning for agile legged locomotion. In *Int. Conf. Automated Planning and Scheduling*, 2018.

[31] Tom Cnops, Zhenyu Gan, and C. David Remy. The basin of attraction for running robots: Fractals, multistep trajectories, and the choice of control. *IEEE International Conference on Intelligent Robots and Systems*, 2015-Decem:1586–1591, 2015.

[32] Leslie Cohen, Thomas F Shipley, Eve Marshark, Kathy That, and Denise Aster. Detecting animals in point-light displays. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 22, 2000.

[33] Steve Collins, Andy Ruina, Russ Tedrake, and Martijn Wisse. Efficient bipedal robots based on passive-dynamic walkers. *Science*, 307(5712):1082–1085, 2005.

[34] Jason Cortell and Pranav Bhounsule. How one might realize practical, energy-efficient legged robots: lessons from the cornell ranger project.

[35] Xingye Da, Zhaoming Xie, David Hoeller, Byron Boots, Animashree Anandkumar, Yuke Zhu, Buck Babich, and Animesh Garg. Learning a contact-adaptive controller for robust, efficient legged locomotion, 2020.

[36] Behnam Dadashzadeh, Hamid Reza Vejdani, and Jonathan Hurst. From template to anchor: A novel control strategy for spring-mass running of bipedal robots. *IEEE International Conference on Intelligent Robots and Systems*, 1(Iros):2566–2571, 2014.

[37] Stefano Dafarra, Sylvain Bertrand, Robert J. Griffin, Giorgio Metta, Daniele Pucci, and Jerry Pratt. Non-linear trajectory optimization for large step-ups: Application to the humanoid robot atlas. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3884–3891, 2020.

[38] Hongkai Dai, Andrés Valenzuela, and Russ Tedrake. Whole-body motion planning with centroidal dynamics and full kinematics. In *2014 IEEE-RAS International Conference on Humanoid Robots*, pages 295–302, 2014.

[39] Monica A. Daley and Andrew A. Biewener. Running over rough terrain reveals limb control for intrinsic stability. *Proceedings of the National Academy of Sciences*, 103(42):15681–15686, 2006.

[40] Jeremy Dao, Kevin Green, Helei Duan, Alan Fern, and Jonathan Hurst. Sim-to-real learning for bipedal locomotion under unsensed dynamic loads. In *2022 International Conference on Robotics and Automation (ICRA)*, 2022.

[41] Jeremy Dao, Kevin Green, Helei Duan, Jonah Siekmann, Yesh Godse, Alan Fern, and Jonathan Hurst. Challenges of Learned High-Speed Locomotion over Five Kilometers in the Real World. In *ICRA 2021: 5th Workshop on Legged Robots: Towards Real-World Deployment of Legged Robots*, 2021.

[42] Robin Deits. Making atlas dance, run, and jump. 6th Workshop on Legged Robots at ICRA 2022, 2022.

[43] Michel C Delfour and J-P Zolésio. *Shapes and geometries: metrics, analysis, differential calculus, and optimization.* SIAM, 2011.

[44] Scott E Dietert. The demonstration of different types of muscle fibers in human extraocular muscle by electron microscopy and cholinesterase staining. *Investigative Ophthalmology & Visual Science*, 4(1):51–63, 1965.

[45] Dimitar Dimitrov, Alexander Sherikov, and Pierre-Brice Wieber. A sparse model predictive control formulation for walking motion generation. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 2292–2299. IEEE, 2011.

[46] Anca D. Dragan, Kenton C.T. Lee, and Siddhartha S. Srinivasa. Legibility and predictability of robot motion. In *2013 8th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 301–308, 2013.

[47] Helei Duan, Jeremy Dao, Kevin Green, Taylor Apgar, Alan Fern, and Jonathan Hurst. Learning task space actions for bipedal locomotion. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1276–1282. IEEE, 2021.

[48] Helei Duan, Ashish Malik, Mohitvishnu S. Gadde, Jeremy Dao, Alan Fern, and Jonathan Hurst. Learning dynamic bipedal walking across stepping stones. In *accepted to 2022 IEEE/RSJ International Conference on Intelligent Robotics and Systems*, 2022.

[49] Michael Ernst, Hartmut Geyer, and Reinhard Blickhan. Spring-Legged Locomotion on Uneven Ground: a Control Approach To Keep the Running Speed Constant. *Mobile Robotics*, pages 639–644, 2009.

[50] Siyuan Feng. Full Body Control for the Atlas robot. *2014 IEEE International Conference on Robotics and Automation*, pages 3733–3738, 2014.

[51] Siyuan Feng, X. Xinjilefu, Christopher G. Atkeson, and Joohyung Kim. Optimization based controller design and implementation for the Atlas robot in the DARPA Robotics Challenge Finals. *IEEE-RAS International Conference on Humanoid Robots*, 2015-Decem:1028–1035, 2015.

[52] Siyuan Feng, X Xinjilefu, Christopher G. Atkeson, and Joohyung Kim. Robust dynamic walking using online foot step optimization. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5373–5378, 2016.

[53] Giorgio Figliolini and Marco Ceccarelli. Climbing stairs with EP-WAR2 biped robot. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*, volume 4, pages 4116–4121. IEEE, 2001.

[54] Michele Focchi, Romeo Orsolino, Marco Camurri, Victor Barasuol, Carlos Mastalli, Darwin G. Caldwell, and Claudio Semini. *Heuristic Planning for Rough Terrain Locomotion in Presence of External Disturbances and Variable Perception Quality*, volume 132, pages 165–209. Springer International Publishing, Cham, 2020.

[55] Zhenyu Gan, Yevgeniy Yesilevskiy, Petr Zaytsev, and C David Remy. All common bipedal gaits emerge from a single passive model. *Journal of The Royal Society Interface*, 15(146):20180455, 2018.

[56] Hartmut Geyer, Andre Seyfarth, and Reinhard Blickhan. Compliant leg behaviour explains basic dynamics of walking and running. *Proceedings of the Royal Society B: Biological Sciences*, 273(1603):2861–2867, 2006.

[57] Yukai Gong and Jessy Grizzle. Angular momentum about the contact point for control of bipedal locomotion: Validation in a lip-based controller. *arXiv preprint arXiv:2008.10763*, 2020.

[58] Kevin Green, Yesh Godse, Jeremy Dao, Ross L Hatton, Alan Fern, and Jonathan Hurst. Learning spring mass locomotion: Guiding policies with a reduced-order model. *IEEE Robotics and Automation Letters*, 6(2):3926–3932, 2021.

[59] Kevin Green, Ross L. Hatton, and Jonathan Hurst. Planning for the unexpected: Explicitly optimizing motions for ground uncertainty in running. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1445–1451, 2020.

[60] J.W. Grizzle, Jonathan Hurst, Benjamin Morris, Hae-Won Park, and Koushil Sreenath. MABEL, a new robotic bipedal walker and runner. In *2009 American Control Conference*, pages 2030–2036. IEEE, 2009.

[61] J-S Gutmann, Masaki Fukuchi, and Masahiro Fujita. Stair climbing for humanoid robots using stereo vision. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, volume 2, pages 1407–1413. IEEE, 2004.

[62] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[63] Roland Hafner, Tim Hertweck, Philipp Klöppner, Michael Bloesch, Michael Neunert, Markus Wulfmeier, Saran Tunyasuvunakool, Nicolas Heess, and Martin A. Riedmiller. Towards general and autonomous learning of core skills: A case study in locomotion. In *4th Conference on Robot Learning, CoRL 2020, 16-18 November 2020, Virtual Event / Cambridge, MA, USA*, volume 155, pages 1084–1099. PMLR, 2020.

[64] Duncan W Haldane, M M Plecnik, Justin K Yim, and Ronald S Fearing. Robotic vertical jumping agility via series-elastic power modulation. *Science Robotics*, 1(1), 2016.

[65] Kaveh Akbari Hamed and Jessy W Grizzle. Event-based stabilization of periodic orbits for underactuated 3-d bipedal robots with left-right symmetry. *IEEE Transactions on Robotics*, 30(2):365–381, 2013.

[66] Matthew L Handford and Manoj Srinivasan. Energy-optimal human walking with feedback-controlled robotic prostheses: a computational study. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 26(9):1773–1782, 2018.

[67] Nicolas Heess, Jonathan J Hunt, Timothy P Lillicrap, and David Silver. Memory-based control with recurrent neural networks, 2015.

[68] Nicolas Heess, Dhruva TB, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, S. M. Ali Eslami, Martin Riedmiller, and David Silver. Emergence of locomotion behaviours in rich environments, 2017.

[69] Steve Heim and Alexander Spröwitz. Beyond basins of attraction: Quantifying robustness of natural dynamics. *IEEE Transactions on Robotics*, 35(4):939–952, 2019.

[70] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[71] At L Hof. Scaling gait data to body size. *Gait & posture*, 3(4):222–223, 1996.

[72] Philip Holmes, Robert J. Full, Dan Koditschek, and John Guckenheimer. The dynamics of legged locomotion: Models, analyses, and challenges. *SIAM Review*, pages 207–304, 2006.

[73] Jerry Pratt and Chee-Meng Chew and Ann Torres and Peter Dilworth and Gill Pratt. Virtual model control: An intuitive approach for bipedal locomotion. *The International Journal of Robotics Research*, 20(2):129–143, 2001.

[74] Christian Hubicki, Jesse Grimes, Mikhail Jones, Daniel Renjewski, Alexander Spröwitz, Andy Abate, and Jonathan Hurst. ATRIAS: Design and validation of a tether-free 3D-capable spring-mass bipedal robot. *International Journal of Robotics Research*, 35(12):1497–1521, 2016.

[75] Christian Hubicki, Mikhail Jones, Monica Daley, and Jonathan Hurst. Do Limit Cycles Matter in the Long Run? Stable Orbits and Sliding-Mass Dynamics Emerge in Task-Optimal Locomotion. *IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[76] Christian M Hubicki and Jonathan W Hurst. Running on Soft Ground: Simple, Energy-Optimal Disturbance Rejection. *Int. Conf. Climbing and Walking Robots and the Support Technologies for Mobile Machines*, pages 543–547, 2012.

[77] Jonathan W Hurst. The electric cable differential leg: A novel design approach for walking and running. *International Journal of Humanoid Robotics*, 8(02):301–321, 2011.

[78] Jonathan W. Hurst, Benjamin Morris, Joel E. Chestnutt, and Alfred A. Rizzi. A policy for open-loop attenuation of disturbance effects caused by uncertain ground properties in running. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 1455–1460, 2007.

[79] Marco Hutter, Mark Hoepflinger, Christian Gehring, Michael Bloesch, C. David Remy, Roland Siegwart, C David Remy, and Roland Siegwart. Hybrid Operational Space Control for Compliant Legged Systems. *Robotics Science and Systems (RSS)*, pages 129–136, 2012.

[80] Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26), 2019.

[81] Brian E. Jackson, Kevin Tracy, and Zachary Manchester. Planning with attitude. *IEEE Robotics and Automation Letters*, 6(3):5658–5664, 2021.

[82] Deepali Jain, Atil Iscen, and Ken Caluwaerts. Hierarchical reinforcement learning for quadruped locomotion, 2019.

[83] Gunnar Johansson. Visual perception of biological motion and a model for its analysis. *Perception and psychophysics*, 14(2):201–211, 1973.

[84] Mikhail S. Jones. *Optimal Control of an Underactuated Bipedal Robot*. Masters of science in mechanical engineering, Oregon State University, 2014.

[85] Shuuji Kajita, Fumio Kanehiro, Kenji Kaneko, Kazuhito Yokoi, and Hirohisa Hirukawa. The 3d linear inverted pendulum mode: A simple modeling for a biped walking pattern generation. In *Proc. 2001 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, volume 1, pages 239–246. IEEE, 2001.

[86] Benjamin Katz, Jared Di Carlo, and Sangbae Kim. Mini cheetah: A platform for pushing the limits of dynamic quadruped control. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6295–6301, 2019.

[87] Benjamin G Katz. A low cost modular actuator for dynamic robots. S.m., Massachusetts Institute of Technology, 2018.

[88] Matthew Kelly. An introduction to trajectory optimization: how to do your own direct collocation. pages 1–44.

[89] Matthew Kelly, Matthew Sheen, and Andy Ruina. Off-line controller design for reliable walking of ranger. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1567–1572. IEEE, 2016.

[90] Gavin Kenneally. Design Principles for a Family of Direct-Drive Legged Robots Design Principles for a Family of Direct-Drive Legged Robots. *IEEE Robotics and Automation Letters*, 1(2):900–907, 2016.

[91] Sang-Hoon Kim. *Electric motor control: DC, AC, and BLDC motors*. Elsevier, 2017.

[92] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[93] Daniel E. Koditschek and Martin Bühler. Analysis of a Simplified Hopping Robot. *The International Journal of Robotics Research*, 10(6):587–605, 12 1991.

[94] Scott Kuindersma. Recent progress on atlas, the world's most dynamic humanoid robot, 2020.

[95] Arthur D Kuo, J Maxwell Donelan, and Andy Ruina. Energetic consequences of walking like an inverted pendulum: step-to-step transitions. *Exercise and sport sciences reviews*, 33(2):88–97, 2005.

[96] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science Robotics*, 5(47):eabc5986, 10 2020.

[97] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[98] Wen-Loong Ma, Ayonga Hereid, Christian M Hubicki, and Aaron D Ames. Efficient hzd gait generation for three-dimensional underactuated humanoid running. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5819–5825. IEEE, 2016.

[99] Yawei Ma and Ren C Luo. Topological method for loop detection of surface intersection problems. *Computer-Aided Design*, 27(11):811–820, 1995.

[100] William C Martin, Albert Wu, and Hartmut Geyer. Experimental evaluation of deadbeat running on the atrias biped. *IEEE Robotics and Automation Letters*, 2(2):1085–1092, 2017.

[101] Jonathan S Matthis and Brett R Fajen. Visual control of foot placement when walking over complex terrain. *Journal of experimental psychology: human perception and performance*, 40(1):106, 2014.

[102] Jonathan Samir Matthis, Jacob L. Yates, and Mary M. Hayhoe. Gaze and the control of foot placement when walking in natural terrain. *Current Biology*, 28(8):1224–1233.e5, 2018.

[103] Seth McCammon, Gilberto Marcon dos Santos, Matthew Frantz, T. P. Welch, Graeme Best, R. Kipp Shearman, Jonathan D. Nash, John A. Barth, Julie A. Adams, and Geoffrey A. Hollinger. Ocean front detection and tracking using a team of heterogeneous marine vehicles. *Journal of Field Robotics*, 38(6):854–881, 2021.

[104] Tad McGeer. Passive dynamic walking. *The International Journal of Robotics Research*, 9(2):62–82, 1990.

[105] Tad McGeer. Passive dynamic biped catalogue, 1991. In *Experimental robotics II*, pages 463–490. Springer, 1993.

[106] Philipp Michel, Joel Chestnutt, Satoshi Kagami, Koichi Nishiwaki, James Kuffner, and Takeo Kanade. GPU-accelerated real-time 3D tracking for humanoid locomotion and stair climbing. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 463–469. IEEE, 2007.

[107] EZ Moore and M Buehler. Stable stair climbing in a simple hexapod robot. Technical report, McGill Research Centre for Intelligent Machines, 2001.

[108] N. Motoi, T. Suzuki, and K. Ohnishi. A bipedal locomotion planning based on virtual linear inverted pendulum mode. *IEEE Transactions on Industrial Electronics*, 56(1):54–61, 2009.

[109] Hae-Won Park, Alireza Ramezani, and Jessy W Grizzle. A finite-state machine for accommodating unexpected large ground-height variations in bipedal robot walking. *IEEE Transactions on Robotics*, 29(2):331–345, 2012.

[110] Hae-Won Park, Patrick M Wensing, and Sangbae Kim. High-speed bounding with the MIT Cheetah 2: Control design and experiments. *The International Journal of Robotics Research*, 36(2):167–192, 2017.

[111] Jeff B. Pelz and Constantin Rothkopf. Chapter 31 - oculomotor behavior in natural and man-made environments. In Roger P.G. Van Gompel, Martin H. Fischer, Wayne S. Murray, and Robin L. Hill, editors, *Eye Movements*, pages 661–676. Elsevier, Oxford, 2007.

[112] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-Real Transfer of Robotic Control with Dynamics Randomization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3803–3810, 2018.

[113] Xue Bin Peng and Michiel van de Panne. Learning Locomotion Skills Using DeepRL: Does the Choice of Action Space Matter? *CoRR*, abs/1611.01055, 2016.

[114] Xue Bin Peng and Michiel van de Panne. Learning locomotion skills using deeprl: Does the choice of action space matter? In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, page 12. ACM, 2017.

[115] Robert R. Playter. *Passive Dynamics in the Control of Gymnastic Maneuvers.* PhD thesis, Massachusetts Institute of Technology, The address of the publisher, 8 1994.

[116] KL Poggensee, MA Sharbafi, and A Seyfarth. Characterizing swing-leg retraction in human locomotion. In *Mobile Service Robotics*, pages 377–384. World Scientific, 2014.

[117] Michael Posa and Russ Tedrake. Direct trajectory optimization of rigid body dynamical systems through contact. In *Algorithmic foundations of robotics X*, pages 527–542. Springer, 2013.

[118] Gill A. Pratt, Matthew M. Williamson, Peter Dillworth, Jerry Pratt, and Anne Wright. Stiffness isn't everything. In *Experimental Robotics IV*, volume 223, pages 253–262. Springer-Verlag, London, 1997.

[119] Jerry Pratt, Peter Dilworth, and Gill Pratt. Virtual model control of a bipedal walking robot. *1997 IEEE International Conference on Robotics and Automation, 1997. Proceedings., 1*, (April):193–198, 1997.

[120] Jerry E Pratt. *Exploiting inherent robustness and natural dynamics in the control of bipedal walking robots.* PhD thesis, Massachusetts Institute of Technology, 2000.

[121] Marc H. Raibert. Legged Robots. *Commun. ACM*, 29(6):499–514, June 1986.

[122] Marc H. Raibert. *Legged Robots That Balance.* Massachusetts Institute of Technology, Cambridge, MA, USA, 1986.

[123] Marc H. Raibert, Jr. Brown, H. Benjamin, Michael Chepponis, Jeff Koechling, Jessica K. Hodgins, Diane Dustman, W. Kevin Brennan, David S. Barrett, Clay M. Thompson, John Daniell Hebert, Woojin Lee, and Lance Borvansky. Dynamically Stable Legged Locomotion. *MIT Technical Report*, (4148):134, 1989.

[124] Marc H Raibert, H Benjamin Brown Jr, Michael Chepponis, Jeff Koechling, and Jessica K Hodgins. Dynamically stable legged locomotion. Technical report, Massachusetts Inst of Tech Cambridge Artificial Intelligence Lab, 1989.

[125] Alireza Ramezani, Jonathan W. Hurst, Kaveh Akbari Hamed, and J. W. Grizzle. Performance Analysis and Feedback Control of ATRIAS, A Three-Dimensional Bipedal Robot. *Journal of Dynamic Systems, Measurement, and Control*, 136(2):021012, 12 2013.

[126] Michael R Rehorn, Alison K Schroer, and Silvia S Blemker. The passive properties of muscle fibers are velocity dependent. *Journal of biomechanics*, 47(3):687–693, 2014.

[127] Daniel Renjewski, Alexander Sprowitz, Andrew Peekema, Mikhail Jones, and Jonathan Hurst. Exciting Engineered Passive Dynamics in a Bipedal Robot. *IEEE Transactions on Robotics*, 31(5):1244–1251, 2015.

[128] Siavash Rezazadeh, Christian Hubicki, Mikhail Jones, Andrew Peekema, Johnathan Van Why, Andy Abate, and Jonathan Hurst. Spring-Mass Walking With ATRIAS in 3D: Robust Gait Control Spanning Zero to 4.3 KPH on a Heavily Underactuated Bipedal Robot. In *Dynamic Systems and Control Conference*, 10 2015. V001T04A003.

[129] David W Robinson. *Design and analysis of series elasticity in closed-loop actuator force control*. PhD thesis, Massachusetts Institute of Technology, 2000.

[130] D.W. Robinson and G.A. Pratt. Force controllable hydro-elastic actuator. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, volume 2, pages 1321–1327. IEEE, 2000.

[131] D.W. Robinson, J.E. Pratt, D.J. Paluska, and G.A. Pratt. Series elastic actuator development for a biomimetic walking robot. *1999 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (Cat. No.99TH8399)*, pages 561–568, 1999.

[132] Agility Robotics. Cassie: Dynamic Planning on Stairs.

[133] Agility Robotics. Agility robotics cassie user manual. `https://github.com/agilityrobotics/cassie-doc/wiki`, 2020. Accessed: 2022-08-14.

[134] Andy Ruina. Cornell ranger, 2011-2012 4-legged bipedal robot. "`http://ruina.tam.cornell.edu/research/topics/locomotion_and_robotics/ranger/Ranger2011/`, 2012.

[135] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

[136] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms, 2017.

[137] Nicola Scianca, Marco Cognetti, Daniele De Simone, Leonardo Lanari, and Giuseppe Oriolo. Intrinsically stable mpc for humanoid gait generation. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pages 601–606, 2016.

[138] Sangok Seok, Albert Wang, Meng Yee Michael Chuah, Dong Jin Hyun, Jongwoo Lee, David M Otten, Jeffrey H Lang, and Sangbae Kim. Design principles for energy-efficient legged locomotion and implementation on the mit cheetah robot. *IEEE/ASME Transactions on Mechatronics*, 20(3):1117–1129, 2015.

[139] André Seyfarth, Hartmut Geyer, and Hugh Herr. Swing-leg retraction: a simple control model for stable running. *Journal of Experimental Biology*, 206(15):2547–2555, 2003.

[140] Mohammad Sharif Shourijeh and John McPhee. Forward dynamic optimization of human gait simulations: a global parameterization approach. *Journal of Computational and Nonlinear Dynamics*, 9(3), 2014.

[141] Jonah Siekmann, Yesh Godse, Alan Fern, and Jonathan Hurst. Sim-to-Real Learning of All Common Bipedal Gaits via Periodic Reward Composition. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[142] Jonah Siekmann, Kevin Green, John Warila, Alan Fern, and Jonathan Hurst. Blind Bipedal Stair Traversal via Sim-to-Real Reinforcement Learning. In *Proceedings of Robotics: Science and Systems*, volume abs/2105.08328, Virtual, 7 2021.

[143] Jonah Siekmann, Srikar Valluri, Jeremy Dao, Lorenzo Bermillo, Helei Duan, Alan Fern, and Jonathan Hurst. Learning memory-based control for human-scale bipedal locomotion. In *Proceedings of Robotics: Science and Systems*, 7 2020.

[144] Youngwoo Sim and Joao Ramos. Tello leg: The study of design principles and metrics for dynamic humanoid robots, 2022.

[145] Nils Smit-Anseeuw, Rodney Gleason, Ram Vasudevan, and C. David Remy. The Energetic Benefit of Robotic Gait Selection—A Case Study on the Robot RAM<italic>one</italic>. *IEEE Robotics and Automation Letters*, 2(2):1124–1131, 2017.

[146] Koushil Sreenath, Hae-Won Park, Ioannis Poulakakis, and Jessy W Grizzle. A compliant hybrid zero dynamics controller for stable, efficient and fast bipedal walking on mabel. *The International Journal of Robotics Research*, 30(9):1170–1193, 2011.

[147] Sarah V Stevenage, Mark S Nixon, and Kate Vince. Visual analysis of gait as a cue to identity. *Applied Cognitive Psychology: The Official Journal of the Society for Applied Research in Memory and Cognition*, 13(6):513–526, 1999.

[148] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[149] Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. In *Proc. of Robotics: Science and Systems XIV*, Pittsburgh, Pennsylvania, 6 2018. Robotics: Science and Systems Foundation.

[150] Russ Tedrake and H Sebastian Seung. Improved Dynamic Stability Using Reinforcement Learning. *5th Int. Conf. on Climbing and Walking Robots (CLAWAR)*, (Figure 1):341–348, 2002.

[151] Travis Teo. From baristas to inspectors: Singapore's robot workforce plugs labour gaps. *Reuters*, May 2022.

[152] Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[153] Matthew Travers, Ross Hatton, and Howie Choset. Minimum perturbation coordinates on so(3). In *2013 American Control Conference*, pages 2006–2012, 2013.

[154] V. Tsounis, M. Alge, J. Lee, F. Farshidian, and M. Hutter. Deepgait: Planning and control of quadrupedal gaits using deep reinforcement learning. *IEEE Robotics and Automation Letters*, 5(2):3699–3706, 2020.

[155] Maegan Tucker, Noel Csomay-Shanklin, Wen-Loong Ma, and Aaron D Ames. Preference-based learning for user-guided hzd gait generation on bipedal walking robots. *arXiv preprint arXiv:2011.05424*, 2020.

[156] Johnathan Van Why, Christian Hubicki, Mikhail Jones, Monica Daley, and Jonathan Hurst. Running into a trap: Numerical design of task-optimal preflex behaviors for delayed disturbance responses. *IEEE International Conference on Intelligent Robots and Systems*, pages 2537–2542, 2014.

[157] H. R. Vejdani, Y. Blum, M. A. Daley, and J. W. Hurst. Bio-inspired swing leg control for spring-mass robots running on ground with unexpected height disturbance. *Bioinspiration and Biomimetics*, 8(4), 2013.

[158] Andreas Wächter and Lorenz T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, Mar 2006.

[159] Patrick Wensing and David Orin. Improved computation of the humanoid centroidal dynamics and application for whole-body control. *International Journal of Humanoid Robotics*, 13:1550039, 09 2015.

[160] Patrick M. Wensing, Albert Wang, Sangok Seok, David Otten, Jeffrey Lang, and Sangbae Kim. Proprioceptive actuator design in the MIT cheetah: Impact mitigation and high-bandwidth physical interaction for dynamic legged robots. *IEEE Transactions on Robotics*, 33(3):509–522, 2017.

[161] Zhaoming Xie, Glen Berseth, Patrick Clary, Jonathan Hurst, and Michiel van de Panne. Feedback control for cassie with deep reinforcement learning. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1241–1246. IEEE, 2018.

[162] Zhaoming Xie, Patrick Clary, Jeremy Dao, Pedro Morais, Jonathan Hurst, and Michiel van de Panne. Learning locomotion skills for cassie: Iterative design and sim-to-real. In *3rd Conf. on Robotic Learning (CORL)*, 2019.

[163] Zhaoming Xie, Xingye Da, Buck Babich, Animesh Garg, and Michiel van de Panne. Glide: Generalizable quadrupedal locomotion in diverse environments with a centroidal model, 2021.

[164] Xiaobin Xiong and Aaron D. Ames. Dynamic and versatile humanoid walking via embedding 3d actuated slip model with hybrid lip based stepping. *IEEE Robotics and Automation Letters*, 5(4):6286–6293, 2020.

[165] William Yang and Michael Posa. Impact invariant control with applications to bipedal locomotion. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5151–5158, 2021.

[166] Yevgeniy Yesilevskiy, Weitao Xi, and C David Remy. A Comparison of Series and Parallel Elasticity in a Monoped Hopper. pages 1–6, 2015.

[167] Haitao Yu, Shengjun Wang, Kaizheng Shan, Jun Li, Lixian Zhang, and Haibo Gao. Seeking the analytical approximation of the stance dynamics of the 3d spring-loaded inverted pendulum model by using perturbation approach. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3314–3320, 2019.

[168] Ali Zamani and Pranav A. Bhounsule. Nonlinear model predictive control of hopping model using approximate step-to-step models for navigation on complex terrain. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3627–3632, 2020.